VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro dynamickou tvorbu formulářů založený na ASP.NET a Windows Azure

System for Dynamic Forms Creation Based on ASP.NET and Windows Azure

2013

Bc. Jakub Raška

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Raška**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Systém pro dynamickou tvorbu formulářů založený na ASP.NET a Windows Azure
System for Dynamic Forms Creation Based on ASP.NET and Windows Azure

Zásady pro vypracování:

Registrace na akce se ukázala během několikaleté zkušenosti při pořádání akcí v rámci IT Academy na naší katedře jako nepostradatelná. Jako ideální pro tyto účely se zdá být systém formulářů v prostředí Google Docs. Systém, který je vytvářen v rámci této práce bude vycházet ze zkušeností s výše uvedeným systémem a dále jej rozšiřovat dle potřeb, které vznikly v rámci IT Academy. Tento systém bude sloužit pro snadnou správu registrací na akce nebo vytváření a vyhodnocování dotazníků. Výsledný systém by měl být finalizován jako krabicové řešení, s přesným konfiguračním postupem pro nasazení na serveru. V rámci této práce provede diplomant implementaci výše uvedeného systému v prostředí ASP.NET s nasazením na Windows Azure a Windows Server 2008.

Jednotlivé cíle diplomové práce jsou:
1. Analýza a návrh systému dynamických formuláře pro organizaci akcí.
2. Prostudování a popis technologie Windows Azure.
3. Prostudování metodik uživatelského testování.
4. Implementace systému.
5. Propojení výstupu systému s kalendáři Google, Windows Live, apod.
6. Provedení uživatelského testování systému a implementace získaných poznatků do systému.
7. Popis systému a jeho nasazení na Windows Azure a na Windows 2008.
8. Testování rozdílu běhu na Windows Azure a Windows 2008 s vyhodnocením získaných dat.
9. Testování zátěže systému a vyhodnocení těchto testů.

Seznam doporučené odborné literatury:

Windows Azure Platform: http://www.microsoft.com/windowsazure/
Kaciho platformový blog: http://blogs.msdn.com/b/kaci/

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 29. dubna 2013 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2013 .....

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Janu Martinovičovi, Ph.D za rady, čas a připomínky, které mi věnoval

Abstrakt

S celosvětovým rozvojem internetových aplikací je dnes žádoucí, aby aplikace měly nejen přístup k datům kdekoliv po celém světě, ale i jejich obsluha byla možná odkudkoliv. Takovýchto aplikací je dnes už mnoho. Tato diplomová práce se zabývá aplikací na vytváření elektronických pozvánek a online dotazníků. Popisuje jak je možné využít cloudovou technologii pro ukládání a zpracovnání dat a porovnává jednotlivá úložiště z hlediska jejich vlastností a rychlosti zpracování dat. V poslední části je popsán návod jak aplikaci připojit na cloud a začít využívat všech jeho výhod.

Klíčová slova: .NET, Windows Azure, ASP.NET

Abstract

With the current global expansion of internet applications, it is desirable for applications to have acces to data no matter where in the world as well as their service available from anywhere. There are many such applications today. This diploma thesis concerns with an application for creating electronic invitations and online questionnaires. It describes the possibilities of using cloud technologies for storing of and manipulation with data and compares individual data storages in light of their properties and speed of manipulation with data. The last part of the thesis provides instructions on how to join an application on cloud and start using all of its benefits.

Keywords: .NET, Windows Azure, ASP.NET

Seznam použitých zkratek a symbolů

HTML	–	Hyper Text Markup Language(jazyk pro tvorbu hypertextových dokumentů)
.NET	–	.NET Framework
ASP.NET	–	Architecture .NET (architektura .NET)
C#	–	C# language(jazyk C#)
VS	–	Visual Studio
VM	–	Virtual machine (virtuální počítač)
URL	–	Uniform resource locator(URL adresa)
XML	–	Extensible markup language(XML jazyk)
SQL	–	Structured Query Language(strukturovaný dotazovací jazyk)
JS	–	JavaScript
IP	–	Internet protocol (Internetový protokol)
SMTP	–	Simple Mail Transfer Protocol (protokol pro zasílání emailový zpráv)

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Elektronická pozvánka na nějakou událost je dnes běžnou věcí. Dnes sice existují webové aplikace, které umožní tvorbu takovéhoto formuláře, ale již nenabízí zpětnou vazbu v podobě přehledu o přihlášených uživatelích, otázkách nebo pozdější korekce formulářů. Z tohoto důvodu byl vytvořen program, který tyto nedostatky napravuje. Umožňuje nejen dynamickou tvorbu, ale i přehled o jednotlivých formlulářích a také přehled o odpovědích jednotlivých účastníků. Aplikace využívá technologii Windows Azure, která umožní, aby program byl neustále dostupný. Tato diplomová práce popisuje takovýto program a jak jej umístit na cloud. Dále se věnuje technologii Window Azure, zejména pak práci s datovými úložišti, která jsou pomocí Windows Azure přístupná a přináší porovnání jejich rychlosti zpracování dat.

1.1 Struktůra práce

Co je to Windows Azure a jaké jsou jeho možnosti je popsáno v kapitole 2. Nedílnou součástí Windows Azure je Compute, která slouží k obsluze výpočetních kapacit cloudu a je popsána v sekci 2.1. Nejvýznamnější službou Windows Azure jsou jeho úložiště (storage), kterým se budeme věnovat v kapitole 2.2. Služba storage má mnoho možností prace s daty. Pomocí blobů zpracováváme nestrukturovaná data (odstavec 2.2.1). Strukturovaná data můžeme zpracovávat pomocí tabulek (odstavec 2.2.2). V odstavci 2.2.3. se budeme zabývat využitím front. Windows Azure může pracovat i s SQL serverem. K této práci uzpůsobil službu SQL Azure (odstavec 2.2.4). Podíváme se taky na možnosti využití úložiště Azure Storage, na které lze přistupovat přímo přes HTTP požadavek (odstavec 2.2.5). Popis lokálního úložiště Windows Azure je v odstavci 2.2.6. V sekci 2.3 a jejich pododstavcích se popisuje jakým způsobem aplikace komunikují v prostředí cloudu a jak se zajišťuje autentizace a autorizace. V sekci 2.4. je popsán Windows Server 2012, který Windows Azure využívá.

Kapitola 3 se zabývá samotným programem pro tvorbu formulářů. Nejprve si představíme program samotný (sekce 3.1) a možnosti bežného uživatele (odstavec 3.1.1 a 3.1.2). V sekci 3.2. jsou pak popsány základní práce s formuláři (odstavec 3.2.1). Pododstavce 3.2.1.1, 3.2.1.2 a 3.2.1.3 se zabývají vytvářením formuláře a jeho vyplněním. Odstavec 3.2.2 popisuje dynamické generování objektů na formulář. Ukládání celých formulářů i s vytvořenými otázkami je pak v odstavci 3.2.3. Sekce 3.3 se zaměřuje na to, jak program pracuje s daty. Práce s SQL Azure je popsána v odstavci 3.3.1. Jak využít v aplikaci blob, jak do něj uložit data a jakým způsobem se nastaví přístupová práva se uvádí v odstavci 3.3.2 až 3.3.5. Manipulace s lokálním úložištěm Windows Azure je vysvětlena v odstavci 3.3.6. Porovnání jednotlivých úložišť, které Windows Azure nabízí je pak v sekci 3.3.7.

V kapitole 4 se zabývám unit testováním a jeho výhodami a nevýhodami (sekce 4.1 a 4.2). Dále popisuju jak testovat ve Visual Studiu - sekce 4.3.

Kapitola 5 se zabývá ICalendářem a jeho jednotlivými častmi. V kapitole 6 je pak sepsán návod jak připravit aplikaci k tomu, aby mohla být umístěna do prostředí cloudu.

2 Windows Azure

Windows Azure je otevřená a flexibilní cloudová platforma, která umožňuje rychlou tvorbu, rozvoj a řízení aplikací po síti, která je tvořena datacentry po celém světě, vytvořené a spravované společností Microsoft[3]. Windows Azure umožňuje vytvářet náročné aplikace bez závislosti na síťové a hardwarové infrastruktuře. Aplikace lze vyvíjet v jakémkoliv programovacím jazyce, nástroji pro vývoj aplikací nebo prostředí. Poskytuje operační systém a jeho údržbu, sestavuje síť a obstarává ochranu proti hardwarovým(dále už jen HW) výpadkům. Je to plně soběstačná platforma, která dokáže zajišťovat výpočetní zdroje v průběhu minut. Platí se pouze za ty zdroje, které přímo aplikace používají.

2.1 Compute

Windows Azure Compute poskytuje výpočetní kapacitu pro aplikace v cloudu. Služba Compute je přístupná prostřednictvím hostovaných služeb, které jsou v podobě balíčků rozmístěny do datových center. Tyto služby poskytují organizační a bezpečné rozhraní pro jednotlivé role. Organizační rozhraní to je proto, že výpočetní zdroje a připojovací metody užívané hostovanými službami jsou specifické pro daný model. Bezpečnostní rozhraní zprostředkovává komunikaci mezi jinými rolemi[2].

Hostovaná služba zahrnuje jednu nebo více rolí. Každá z nich poskytuje specifickou funkcionalitu dané služby. Po spuštění běží alespoň dvě instance téže role, přičemž každá role pracuje jako samostatný virtuální počítač. Všechny instance rolí mají stejnou velikost specifikovanou v modelu služby. Avšak různé typy rolí mohou mít různé velikosti. Síla cloud computingu nepochází z velikosti jednotlivých instancí, což je ze své podstaty omezené, ale z různého počtu možných instancí. Zatímco velikost role je definována při jejím vzniku, počet instanci je variabilní a záleží na potřebách aplikace. Ve Windows Azure máme 3 typy rolí:

- **Web role** je přizpůsobena pro webové aplikace, které podporují Internet Information Services 7 (dále jen IIS) a ASP.NET. Výhoda této role spočívá v již nastaveném IIS. Role je ideální pro webové aplikace nebo hostované služby.
- **Worker role** musí být spuštěna nějakým skriptem. Je proto vhodná pro aplikace, u kterých je potřeba pro spuštění nějaký impulz. Může běžet na pozadí Web role a je vhodná pro dlouho trvající procesy.
- **Virtual machine role** nabízí nejvyšší flexibilitu, ale za cenu vysoké pracnosti. V této roli dodáváme celý virtuální disk příslušného serveru, o který se musíme starat. Musí se instalovat opravy, provádět konfigurace. Tato role je bezestavová. Stav virtuální role není zálohován, pokud dojde k hardwarové poruše, role se uvede do stavu, kdy byla instalována. Použití této role ale není časté a většina běžných aplikací lze vyřešit použitím web nebo worker role.

2.2 Storage

Azure Storage poskytuje zabezpečené, dostupné a lehce přístupné úložiště, které je stálé. Azure Storage služba podporuje doslova všechny typy úložišť, které potřebujeme. Můžeme ukládat jak strukturovaná data, tak i nestrukturovaná, NoSQL databáze a fronty[4]. Služba Azure Storage nám zpřístupňuje metody a objekty, pomocí kterých můžeme ukládat nebo načítat data. Z důvodu bezpečnosti jsou data replikována hned 3x. V případě výpadku jedné z replik se data začnou načítat ze zbylých replik a ihned se vytváří replika nová. Jednotlivý Windows Azure účet může pojmout až 100 TB dat. Pokud uživatel potřebuje ukládat větší množství, musí si vytvořit nový účet. V rámci jednoho účtu je možné ukládat data do blobu, tabulek nebo front.

2.2.1 Blob

Blob reprezentuje jednoduchý způsob ukládání velkého množství nestrukturovaných nebo binárních dat, jako jsou například dokumenty, obrázky, videa nebo audio. Datové úložiště vlastní kontejner, do kterého se jednotlivé bloby ukládají. Kontejnery se chovají podobně jako adresáře. Shromažďuji uvnitř sebe data a tím je oddělují od dalších dat v ostatních kontejnerech. Existují dva druhy blobů: Block Blob a Page Blob. K oběma typům blobů můžeme přistupovat i přes jednoduchý http požadavek a data jednoduše číst. Bloby mohou být nastaveny jen pro čtení, což z nich dělá ideální způsob pro ukládání fotografií, videí a podobně. Pomoci služby Shared Acces Signature (SAS) lze jednotlivým blobům nastavit možnost, že uživatelé nebudou potřebovat znát přístupové údaje k účtu, aby data mohli mazat nebo je jiným způsobem upravovat.

- **Block Blob** ukládá do jednotlivých bloků. Každý blok je identifikován klíčem a může nabývat velikosti až 4 MB. Maximální velikost celého blobu je pak omezena na 200 MB. Je to ideální způsob pro většinu scénářů, které uživatel bude používat. Umožňuje jednoduché vkládání, mazání nebo přeuspořádání bloku dat.
- **Page Blob** je vytvořen pro náhodně read/write scénáře. U tohoto blobu můžu kdykoliv sáhnout na jakékoliv místo uvnitř blobu a číst nebo zapisovat data. Limit blobu je až 1T. Je ale náročnější na režii, protože musíme někde zaznamenávat indexaci. Tento typ blobu se používá nejčastěji pro virtuální disky. Ke každému blobu je možné přiřadit metadata, která můžou sloužit jako doprovodná informace nebo identifikace jednotlivého blobu.

2.2.2 Table

Tabulky slouží k nerelačnímu ukládání strukturovaných dat, která lze popsat tabulkou. Mohou to být informace o zákaznicích, objednávkách, zpravodajství a podobně. Tabulka obsahuje entity. Na rozdíl od klasické databázové tabulky jednotlivé entity mohou mít v rámci jedné tabulky rozdílné vlastnosti. Například do jedné tabulky mohu uložit jak zákazníka, tak i objednávku. Atributy Row Key a Partition Key dohromady identifikuji jednotlivé entity uvnitř tabulky. To znamená, že není možné, aby dvě různé entity měly

stejný Partition Key a Row Key v rámci jedné tabulky. Entity v tabulce se setřizují nejprve podle Partition Key. Ty entity, které mají stejný Partition Key, jsou dále setříděny podle Row Key. Služba Table Storage podporuje základní vytváření, čtení, úpravu a mazání dat. Na rozdíl od klasických databázových tabulek, Table Storage nepodporuje úkony jako spojování tabulek, cizí klíče, popřípadě uložené procedury. Pokud tyto úkony budeme chtít nad našimi daty provádět, je lepší použít službu SQL Azure, která tyto úkony podporuje.

2.2.3 Qeue

Azure Queue - fronta úloh, které má softwarová komponenta zpracovávat. Používá se na komunikaci pomocí zpráv tak, aby byla zajištěna vysoká škálovatelnost aplikace. Obsahuje dvě komponenty. Jedna práci zadává, druhá ji vykonává. První je nějaké uživatelské rozhraní, ta druhá je nějaká business logika, která zprávy zpracovává. Výhodou tohoto přístupu je, že umožňuje rozkládat špičky. Úlohy jsou ve frontě a servery je zpracovávají podle toho jak stíhají. Zprávy mohou byt maximálně 64 kB. Není zaručeno, že zprávy budou zpracovány v přesném pořadí.

2.2.4 SQL Azure

SQL Azure poskytuje vysoce dostupný a škálovatelný relační databázový systém pro Windows Azure. Výhodou tohoto systému je, že je kompatibilní s SQL Serverem, který umožňuje užívat známé T-SQL, SQL knihovny a podobně. Například použitím ADO.NET nebo ODBC umožní práci s SQL Azure aniž bychom příliš měnili kód. SQL Azure je ideální nástroj pro práci s daty, nad kterými chceme provádět operace typu spojování, třídění, používání uložených procedur. Pro přístup do databáze používá port 1433.

SQL Azure funguje jako velké množství virtuálních počítačů, na kterých je velké množství databází, ke kterým se uživatel připojuje. Uživatel nemusí vědět jak to fyzicky funguje. Veškerá data jsou replikována. Logicky vidím jen jednu databázi, ale ve skutečnosti jsou za ní 3 repliky. Jedna je primární, dvě záložní. Každá transakce je zapsána aspoň do dvou replik. Uživatel se připojuje pouze do jedné databáze, ostatní jsou tam jen z důvodu redundance. Když dojde k selhání, tak se automaticky vytváří další replika a jedna z přeživších replik je zvolena jako primární. Aplikace o tomto procesu netuší.

2.2.5 Azure storage

Azure Storage je alternativní relační přístup k uložení relačních dat. Data nejsou přístupná přes T-SQL, jsou přístupná přes rozhraní rest, což je vlastně http protokol, nebo linq přístup. Je vhodná pro uložení velkého objemu dat. Knihovny jsou dostupné na téměř jakoukoliv programovací platformu. K datům je možno přistupovat přímo. Zde taky můžeme použít uložení do tabulek, získáme tím prakticky neomezenou škálovatelnost a velikost DB. Data jsou ukládána do takzvaných oddílů (particions). Jakýkoliv řádek, který ukládáme do Azure Storage, musíme označit partition klíčem a row klíčem. Čím více oddílů, tím větší škálovatelnost a tím větší možnost rozdělit data mezi více serverů.

Na druhou stranu transakce nebo dotazy nelze provádět napříč oddíly, je proto nutné najit nějaký kompromis. Azure Storage je spíše výhodný pro aplikace, které data spíše ukládají a dotazují se méně. SQL server je vhodný pro databázi, kde se data málo ukládají, ale probíhá nad nimi jeden dotaz za druhým. Výhoda obou technologií je bezesporu v pořizovacích nákladech. Odpadají náklady na pořízení hardware a jeho údržbu. Platíme jen za provoz.

2.2.6 Local storage

Vyhrazená část lokálního souborového systému, která je vyhrazena pro potřeby aplikace. Je to vlastně místo na disku virtuálního počítače, do kterého mohu ukládat jakákoliv data. Jedna se o úložiště privátní. To znamená, že jednotlivé role vidí a mohou pracovat jen se svým lokálním úložištěm a nemají přístup na úložiště jiných rolí. Local Storage je úložiště neperzistentní, to znamená, že data nejsou nijak zálohována nebo replikována a tím pádem data, která tam jsou uložena, mohou byt kdykoliv ztracena z důvodu například hardwarové havárie. Lokální úložiště je tedy použitelné pro data, která potřebujeme uložit na krátkou dobu, například různé mezivýsledky, logovací záznamy a podobně.

2.3 Conectivity

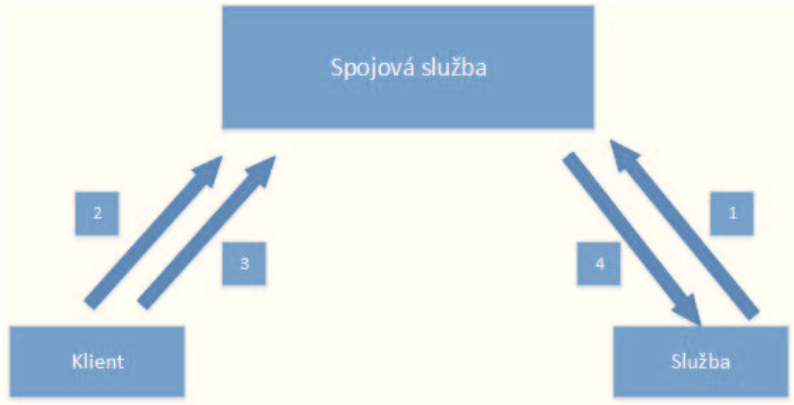
Technologie App Fabric umožňuje aplikacím vzájemnou komunikaci v prostředí cloudu a to i do firemních sítí, které často bývají schovány za různými firewally nebo NAT adresami. Aplikace nemusí být pouze na platformách Windows Azure nebo Windows Server. App Fabric můžeme použít i pro platformy jako Java, Ruby, PHP atd. Mezi významné služby App Fabric patří například Service bus, Access control.

2.3.1 AppFabrtic Service bus

Služba .NET Service Bus je pravděpodobně nejužitečnější a nejvýznamnější službou Windows Azure. Tato služba byla navržena nejen na překonání problémů s připojením, ale poskytuje také řešení problémů škálovatelnosti, dostupnosti a bezpečnosti[1].

Spojová služba podporuje přímou jednosměrnou komunikaci, request/responce komunikaci a peer to peer komunikaci. Spojová služba umožňuje aplikacím běžícím na Windows Azure bezpečně komunikovat s aplikacemi na cloudu. Je to jakýsi prostředník mezi oběma aplikacemi, který spojení zprostředkovává. Při používání spojové služby není třeba iniciovat a nastavovat nove spojeni pro každý požadavek. Vytvořené spojení je tedy rychlé a bezpečné. Spojová služba má schopnost spojit aplikace přes existující NAT a firewally. Spojová služba podporuje různé transportní protokoly a webové standardy včetně REST, SOAP.

Jak je možné vidět na obrázku 2, nejprve musí klient i koncová služba navázat spojení s hostovanou službou. Hostovaná služba zároveň zjišťuje kde koncová služba je a zjistí nejlepší postup jak navázat komunikaci. Když klient vyšle požadavek k hostované službě, je okamžitě přesměrován na koncovou službu. Klient má pocit, že spojení je přímé bez jakéhokoliv prostředníka.



Obrázek 1: Schéma spojové služby

2.3.2 Windows azure connect

Pomocí Windows Azure Connect [5] můžeme používat jednoduchý interface ke konfiguraci chráněné komunikace mezi počítačem nebo serverem v naší organizační síti a rolemi běžícími ve Windows Azure. Windows Azure Connect je založen na protokolu IPsec. Pokud jde o funkčnost, Azure Connect zprostředkovává jakýsi most mezi jednotlivými rolemi a službami běžícími jak v cloudu, tak u uživatele. Azure Connect je méně zaměřený na meziresortní komunikaci než Service Bus. Windows Azure Connect vytvaří logickou síť, která může obsahovat dva typy entit:

- **Azure role group** mapuje role, které byly vytvořeny pro Access Control Service. Jedině Azure VM instance pro role jsou členy této skupiny rolí. Administrátor nemůže manuálně přidávat nebo odebírat členy. Access control service automaticky řídí členství ve skupině. Jestliže se přidá nebo odebere instance, role Access control service to zaznamená a upraví skupinu rolí.
- **Azure machine group** jsou administrátorem definované skupiny externích počítačů, které byly vytvořeny pro Access control service přes instalaci Windows Azure Connect endpoint software. Externí zařízeni může náležet maximálně jedné skupině.

2.3.3 Access control service

Windows Azure Access Control Service (ACS) [6] je cloudová služba, která poskytuje jednoduchý způsob jak autentizovat a autorizovat uživatele snažící se získat přístup do našich webových aplikací a služeb. Vývojáři aplikace dává nástroje k tomu, aby metody autentizace a autorizace mohly být zpracovány přímo v kódu aplikace. V aplikaci spravované ACS není nutné implementovat autentizační systém s uživatelskými účty, který je specifický pro vaši aplikaci, ale můžeme nechat organizovat autentizaci a velkou část autorizace uživatelů právě na ACS. ACS v sobě zahrnuje identifikační standardy včetně

podnikových adresářů, jako je Active Directory a internetových identit jako například Windows Live ID, Google nebo Facebook. Uživatelé, kteří přistupují na stránku aplikace jsou ACS vyzváni, aby si zvolili identitu, u které mají účet. ACS uživatele přesměruje na přihlašovací stránku a po úspěšném přihlášení přesměruje zpět na stránku aplikace. Vývojář, který vyvíjí aplikaci spravovanou ACS si může zvolit, kterým identitám bude ACS důvěřovat. Nezáleží pak na tom z jakého zdroje přihlášení proběhlo. Aplikace věří ACS a pokud bude tvrzení o přihlášení ACS ověřeno, má uživatel povolen přístup.

2.4 Windows server 2012

Windows Server 2012 je šestý v řadě Windows Serverů od společnosti Microsoft. Přináší řadu vylepšení a nových nástrojů, které umožňují snížit náklady a zvýšit efektivitu aplikací umístěné na cloud. Windows Server 2012 byl navržen od základu přímo pro práci s cloudem. [7]

2.4.1 Network virtualization

Cloud poskytuje k provozu datacentra, která nabízejí virtuální počítače k pronájmu, spolu s dynamicky přidělovanými výpočetními zdroji. Zákazník vlastní virtuální počítač a provozuje jej jako server v cloudu. Mohou nastat dva scénáře užití.

- **Ve scénáři share private** cloud poskytuje svoje vlastní uspořádání. Vlastní a provozuje své datové centrum, přičemž zákazníci mohou mít různé lokace, odkud se připojují.
- **Scénář share public** je cloud poskytovatel hostingu. Zákazníci mohou být velké korporace, střední firmy nebo malé podniky. Hostující společnosti vlastní a spravují datová centra a mohou pronajímat servery zákazníkům.

Network virtualization je nová vlastnost ve Windows Serveru 2012, která umožní udržet si vlastní IP adresu i v době přesunu na servery cloudu. Network virtualization umožňuje přiřadit dva druhy IP adres každému virtuálnímu počítači, který pracuje pod Windows Server 2012. Tyto aresy jsou

- **Customer address (CA)** Tuto IP adresu měl zákazník před tím než svou aplikaci přesunul na cloud.
- **Provider address (PA)** Tuto adresu přiděluje samotný cloud a může se přidělit v době, kdy aplikace byla přemístěna na cloud.

Z pohledu zákazníka je komunikace s přesunutým serverem stejná, jako když aplikace běžela na jeho vlastních serverech. To proto, že virtuální počítač používá zákazníkovu adresu, kterou si přenesl spolu s aplikací. Virtuální počítač nemůže vidět nebo používat zákazníkovu adresu, protože tato adresa je viditelná pouze pro provozovatele serveru.

Síťová virtualizace tak umožňuje poskytovateli cloudu provozovat několik virtuálních sítí na fyzické vrstvě v podstatě stejným způsobem jako serverová virtualizace umožňuje

běh několika virtuálních serverů na jednom fyzickém serveru. Síťová vizualizace také izoluje každou virtuální síť od druhé s tím výsledkem, že každá síť se domnívá, že tam je sama. To znamená, že dvě nebo více virtuálních sítí mohou mít stejné adresování, přesto sítě budou na sobě nezávislé a jedna od druhé oddělena.

Aby toto všechno bylo možné, síťová vizualizace potřebuje způsob jak virtualizovat Ip adresy a následně je namapovat na fyzickou vrstvu. Síťová vizualizace ve Windows serveru 2012 nabízí dva způsoby jak toho docílit

- **Network Virtualization Generic Routing Encapsulation (NVGRE)** V tomto případě všechny pakety virtuálních počítačů jsou zapouzdřeny s novou hlavičkou předtím, než jsou přeposlány na fyzickou vrstvu. NVGRE potřebuje pouze jeden PA na účastníka, který je sdílen všemi virtuálními počítači pro daného poskytovatele.
- **IP rewrite** Tento postup upravuje zákazníkovy adresy paketů ještě v době kdy jsou ve virtuálním počítači a předtím než jsou přeposlány na fyzickou vrstvu. IP rewrite potřebuje one-to-one mapování zákaznických adres na adresy poskytovatele.

2.4.1.1 Výhody

Síťová virtualizace je klíčem k tomu, abychom mohli vytvářet cloudové služby pro více různých uživatelů. Ať už to jsou velké společnosti nebo menší firmy či jednotlivci, síťová virtualizace nabízí možnost vytvořit mnoho sítí, které jsou jedna od druhé izolovány a to vše bez omezení nebo nároků spojených s vytvářením VLAN sítí.

2.4.2 Hyper-V extensible switch

Hyper-V extensible switch ve Windowsu 2012 je klíč k umožnění tvorby bezpečného cloudového prostředí, které pomáhá oddělit jednotlivé uživatele cloudu. Extensible switch umožňuje třetím stranám vyvíjet plug-in rozšíření k dosažení úplné podpory hardwarových přepínačů a větší podpory komplexního virtuálního prostředí a řešení.

Dřívější verze Hyper-V umožňovala implementaci komplexního virtuálního síťového prostředí, tvořeného virtuální sítí přepínačů, které pracovaly jako přepínače fyzické vrstvy. Mohla se tak vytvořit vnější virtuální síť k poskytování virtuálních počítačů, které komunikovaly s uživateli a servery. Jednotlivé virtuální počítače bylo možné také od sebe izolovat a provádět komunikaci pouze prostřednictvím externí sítě.

Hyper-V extensible switch usnadňuje vytváření virtuálních sítí, které mohou být implementovány mnoha způsoby k poskytnutí co největší flexibility ve způsobu jak navrhnout jakoukoliv virtuální infrastrukturu. Například můžeme nastavit operační systém uvnitř virtuálního počítače a mít jednoduchý virtuální adaptér spojený s konkrétním extensible přepínačem nebo několikanásobný virtuální síťový adaptér (každý spojený s jiným switchem), ale už není možné připojit jeden switch k několikanásobnému adaptéru.

2.4.3 Živá migrace

Live migration byla představena už ve Windows Serveru 2008 R2, aby poskytovala řešení pro virtuální počítače, které poskytují hosting. Live migration používá Failover Cluste-

ring, která umožňuje přesun virtuálních počítačů mezi uzly sítě, aniž by došlo k přerušení spojení nebo k nějakému prodlení. Live migration přináší výhodu vyšší schopnosti přesunu virtuálního počítače na lepší místo pro zvýšení výkonu, dosažení lepší škálovatelnosti, nebo optimálního vytížení.

Live migration na Windows Serveru 2012 byla oproti Windows Serveru 2008 R2 vylepšena hned několikrát.

Live migraci lze provádět mnohem rychleji. Vlastně můžeme dokonce využít až 10 GB síťového připojení. Tato možnost na Windows Serveru 2008 R2 nebyla.

Druhé zlepšení spočívá v tom, že můžeme provádět vice migrací současně v rámci jednoho uzlu sítě. To znamená, že například pokud potřebujeme vypnout nějaký uzel, at už z důvodu údržby nebo něčeho jiného, můžeme jednoduše přesunout všechny virtuální počítače z tohoto uzlu do jiného najednou. Tato schopnost výrazně urychlí údržbu v rámci cloudu.

Poslední zlepšení je, že live migrace je nyní možná přestože nemáme nasazenou službu failover clustering. V předchozí verzi Windows Serveru live migrace potřebovala instalování Failover clustering funkci a bylo nutné zajistit, aby úložiště pro Cluster Sharp Volume bylo přístupné a aby virtuální počítač měl k tomuto úložišti přístup v daném okamžiku. S Windows Server 2012 máme dvě možnosti live migrace.

- **Můžeme uložit virtuální počítač do sdílené složky v sítí** ,která nám umožní živou migraci mezi ne Clusterovými Hyper-V hostiteli, zatímco zbytek virtuálního počítače je ve sdílené složce.
- **Můžeme také použít živou migraci** přímo mezi dvěma Hyper-V hostiteli bez použití sdíleného úložiště.

3 Program na tvorbu dynamických formulářů

3.1 Popis programu

Aplikace je napsána v jazyce ASP.NET a umožňuje vytvářet formuláře, které mohou být použity buďto jako přihláška na nějakou událost(přednáška, prezentace atp.), nebo jen jako dotazovací formulář. Ke každému formuláři lze vytvořit sadu otázek, na které účastník, pokud se chce přihlásit na událost, musí odpovědět. Zadavatel má přehled o svých vytvořených formulářích a odpovědích jednotlivých uživatelů. V případě dotazníku lze odpovídat i anonymně. S každým formulářem se vygeneruje kalendářní soubor, který může uživatel používat ve své kalendářní aplikaci. Aby někdo mohl daný formulář vyplnit, musí být přihlášen do systému. Tímto přihlášením získává možnost přehledu všech svých událostí, na které se přihlašoval a také svých odpovědí.

3.1.1 Přihlašovací formulář

Formulář tvoří základní komunikaci mezi aplikací a těmi, kteří se chtějí na událost přihlásit nebo chtějí někoho na událost pozvat. Kromě základních údajů jako je místo konání nebo čas, je možné, aby zadavatel akce vytvořil sadu otázek, na které pak účastníci musí odpovědět a získal tak zpětnou vazbu prostřednictvím jejich případných postřehů či připomínek. Každý formulář je identifikován pomocí klíče. Hashovací klíč je řetězec, který je tvořen 24 náhodně vygenerovanými znaky. Abychom zamezili generaci dvou stejných klíčů, je před každý takovýto řetězec ještě přidáno identifikační číslo, které označuje řádek v tabulce kde je záznam o formuláři uložen. Tímto je zajištěno, že žádný hashovaní klíč nebude nikdy stejný. Takovýto klíč neslouží jen k identifikaci formuláře, ale i k identifikaci souboru s kalendářem, který je vygenerován s každým formulářem. Formulář může vytvářet každý, kdo je přihlášen do systému. Formuláře jsou uloženy v tabulce FormTable, která je vidět v příloze B.

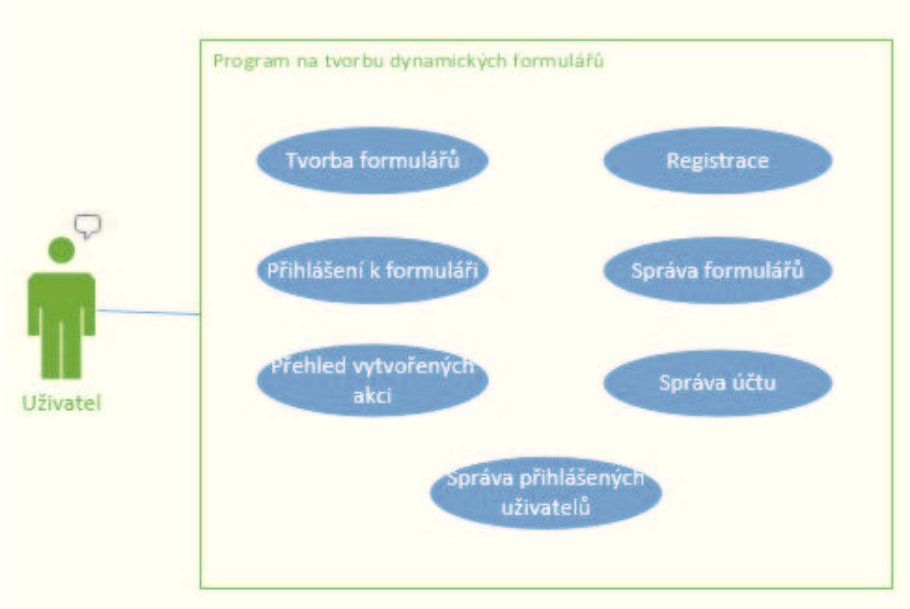
3.1.2 Role uživatel

Tento typ role je nejrozšířenější a je jím kdokoliv, kdo se registruje do systému. Přihlášením se zpřístupní základní funkce programu jako je přihlašování a vyplňování formulářů, jejich samotné vytváření a následná editace nebo přehled již vytvořených formulářů.

3.1.2.1 Přehled formulářů Jednotlivé události nebo dotazníky jsou vypsány v nabídce menu Mé akce. Jak je možné vidět na obrázku 3, každý uživatel má zobrazeny své formuláře a může se podívat nejenom na přihlášené uživatele (obrázek 4).

Výpis přihlašovacích informací mezi něž patří například vygenerovaný kalendářní soubor nebo HTML kód k přihlašovacímu Iframe, na kterém jsou vyzobranzeny otázky k danému formuláři (obrázek 5).

Pomocí nabídky detail můžeme editovat jednotlivé informace ve formuláři. Od změny datumu či místa konání až po otázky pro přihlašující se a vzhled iframu.



Obrázek 2: Schéma možností uživatele

Obrázek 3: Uživatelské menu

3.1.2.2 Editace formulářů Jak už bylo zmíněno výše, každou událost je možné zpětně editovat, at'už se jedná o změnu vzhledu, data konání, nebo úplné přepsání dotazovacího formuláře. Po kliknuti na nabídku detail se nám zobrazí potvrzovací okno klasického formuláře (obrázek 6).

Z tohoto formuláře lze přejít na úpravu místa a času konání nebo na úpravu otázek pro formulář. Krok na úpravu otázek nám zpřístupní take úpravu vzhledu.

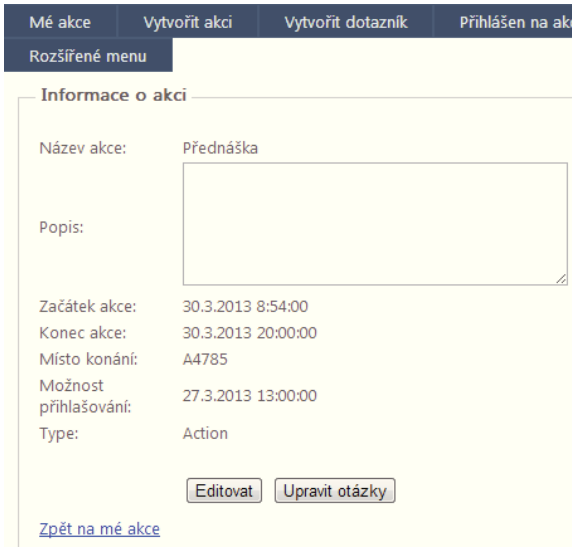
Uživatel má taky možnost výpisu jednotlivých odpovědí na jednotlivé otázky, at' už formou stručného přehledu nebo podrobnějšího výpisu, kde jsou sečteny jednotlivé odpovědi na každou otázku samostatně. Přihlašovací informace, jako je datum konání, místo a čas, spolu s přihlášenými uživateli lze vyexportovat do PDF nebo Excel souboru.

Obrázek 4: Přihlášení uživatelé

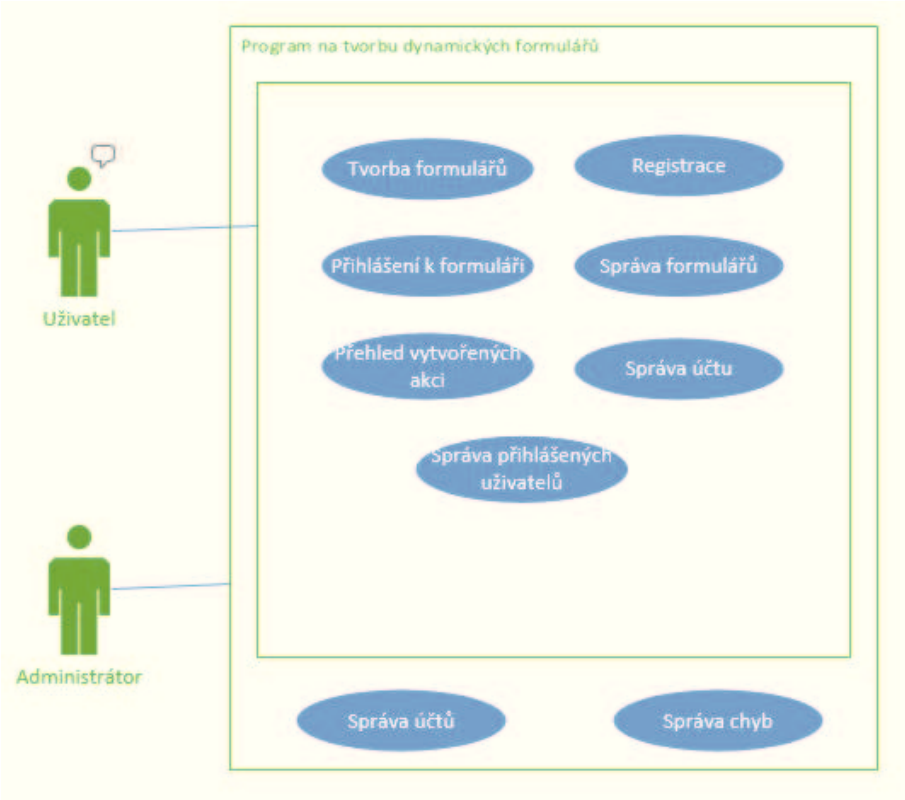
Obrázek 5: Informace o události

3.1.3 Role administrátor

Role administrátor sdílí všechny funkce, které má k dispozici běžný uživatel, takže může nejenom vytvářet akce a pote je spravovat, ale má možnost i přihlásit se na libovolnou událost nebo zodpovědět dotazník. Oproti běžnému uživateli má role administrátor možnost správy jiných účtů (obrázek 8).



Obrázek 6: Úprava formuláře



Obrázek 7: Schéma možností administrátora

ID	login	Name	Surename	mail	phone	Role		Login	
1	non	non	non	non	non	1	Detail	Přihlášení uživatelé	Jejich akce
3	tester	tester	testerovic	g.jamezz@gmail.com	777	1	Detail	Přihlášení uživatelé	Jejich akce

Obrázek 8: Přehled uživatelů

Hlavní funkcí administrátora ovšem je přehled chyb, které aplikaci nějakým způsobem znemožnily chod. Všechny tyto chyby se logují do databáze, kde má administrátor přehled o jejich původu a důvodu. Jedinou výjimkou jsou chyby, které zapříčinila nefunkční databáze. Tyto chyby se zapisují do xml souboru. Chyby programu se zachytávají v souboru Global.asax. Tento soubor obsahuje metodu Application_Error a pokud se vyskytne nějaká chyba pomocí této metody se zachytí a uloží. Jednotlivé chyby si pak administrátor může vypsat v přehledné tabulce a má tak možnost reagovat na případné pády aplikace (obrázek 9). Celý zdrojový kód lze nalézt v příloze A.

Obrázek 9: Přehled chyb aplikace

Po kliknutí na Select, se zobrazí administrátorovi podrobný výpis o chybě i přesném umístění. Ukázka je zobrazena v příloze C

3.2 Základní funkce

Mezi základní funkce aplikace patří tvorba formulářů, kdy uživatel vyplní místo a datum konání. Nastaví, do kdy se mohou zájemci přihlašovat, nebo do kdy lze na dotazník odpovědět. Sestaví seznam otázek a určí vzhled jakým se formulář zobrazí na stránce, kde se umístí. Pomocí takto umístěného formuláře se pak jednotliví účastníci přihlašují.

3.2.1 Struktůra formuláře

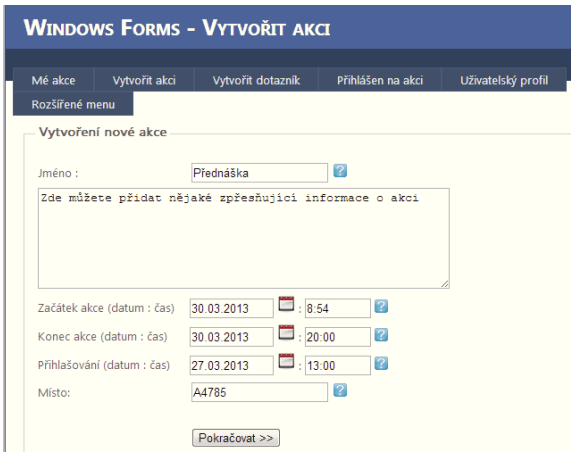
Samotná struktura každého formuláře se skládá ze šesti tabulek. Čtyři tabulky slouží k ukládání informací nutné pro zpracovávaní formuláře. Zbylé dvě jsou vazební.

- **FormTable** obsahuje informace o jednotlivých formulářích, jako například datum a čas konání, název akce.

- **UserTable** ukládá data o uživatelích, kteří jsou zaregistrováni v systému.
- **QueryTable** definuje otázky k jednotlivým formulářům. Jsou zde uloženy informace o typu otázky a její konkrétní znění.
- **Style** tabulka definuje grafické nastylování formuláře, které se bude uživateli zobrazovat.
- **BtOwnerForm** je vazební tabulka a určuje, kteří uživatelé jsou přihlášení k jednotlivým formulářům.
- **BtFormUser** je také vazební tabulka a určuje konkrétního vlastníka daného formuláře.

3.2.2 Tvorba formulářů

Formulář se tvoří pomocí tří na sebe navazujících kroků. První krok je založení události (obrázek 10). Jedná se o popis místa a času konané akce. Zadá se údaj od kdy do kdy je možné se přihlašovat. Zvolí se zde, zda jde o událost na kterou mohou lidé dorazit a zúčastnit se, nebo se jedna o formu dotazníku, který jen uživatel vyplní. Druhá část je



Obrázek 10: Vytvoření akce

tvorba otázek. Uživatel má na výběr z 5 možných typů otázek. 3 typy jsou otázky, kdy návštěvník vybírá z předem definovaných možností. Buďto je možná jedna odpověď nebo víceznačná odpověď. Poslední dva typy jsou odpovědi, které uživatel vpisuje. Otázky lze různě kombinovat, může mít libovolný počet možností a tím si tvůrce formuláře může plně otázky přizpůsobit jak uzná za vhodné. Každá otázka musí mít text, kterým se dotazujeme a nějakou možnost odpovědi z výše uvedených typů. K otázce je možné přidat i vysvětlující text, kterým může více přiblížit, co se v otázce požaduje. Poslední 3. část je stylování formuláře. Ke každému formuláři je přiřazena tabulka Style. Pomocí této tabulky můžeme nadefinovat vzhled formuláře tak, aby se snáze začlenila do vzhledu

stránky, kde bude formulář umístěn. Po splnění všech tří kroků se formulář uloží do databáze. Poté co je formulář vytvořen, vygeneruje se kalendář, který obsahuje námi vytvořenou akci. K tomuto kalendáři je možný přistup pomocí http požadavku. [1]

Je také vygenerován HTML kód iframu, který si můžeme umístit na stránku. Pokud nechceme nebo nemáme možnost takovýto iframe nikam umístit, můžeme využít prostředí aplikace a přihlásit se prostřednictvím aplikace. [2] S potřebnými informacemi je také přiložen odkaz na návod, jak importovat ICalendar do prostředí služby HotMail.

3.2.2.1 Vyplnění formulářů Přihlášení probíhá pomocí Iframe. Tento Iframe si může kdokoliv umístit do své stránky a uživatelé se mohou pomocí něj přihlašovat. Iframe nejdříve vyzve uživatele, aby se přihlásil do systému, čímž tak ověřil totožnost přihlašovaného. V iframu jsou zobrazeny informace o události, které informují o tom, kdy se daná akce koná, na jakém místě a popis dané události.

```
<IFRAME src=http://mythesis.cloudapp.net/LoginIframe.aspx?hashkey=10085
    T1h3uD9aJwDB400EN0ohfhj width=600 height=650></IFRAME>
```

Výpis 1: Přiklad vygenerovaného Iframe

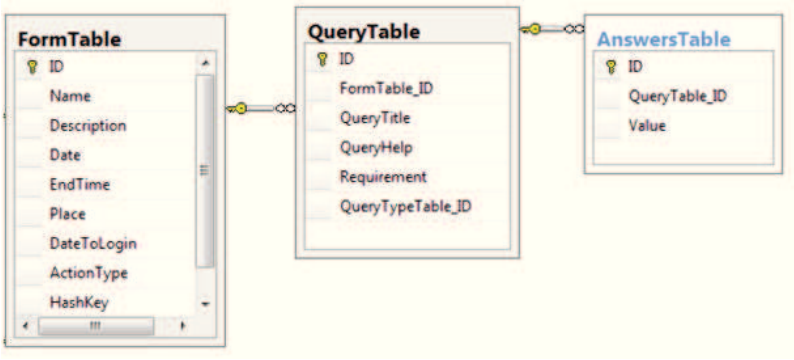
V kódu Iframe jsou definovány rozměry a identifikační klíč, který určuje, o který formulář se jedná a také který ICalendář bude pak přihlášený uživatel moci používat. K vyplnění dat jsou zapotřebí 3 tabulky (obrázek 11), které určí ke kterému formuláři náleží jaká odpověď.

- **UserAnswerTable** zaznamenává odpovědi uživatelů na konkrétní otázku formuláře.
- **QueryTable** definuje otázky k jednotlivým formulářům. Jsou zde uloženy informace o typu otázky a jeji konkrétní znění.
- **UserTable** ke každému uživateli přiřadí konkrétní odpověď.

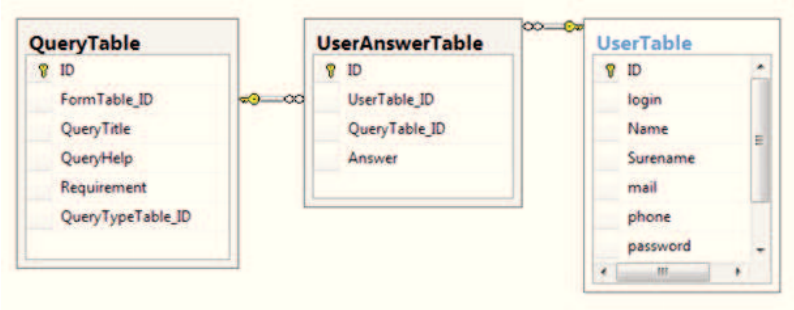
3.2.2.2 Vyplnění dat Poté co proběhne ověření identity uživatele, může přejít na samotné přihlášení na událost. Tato akce spočívá ve vyplnění otázek, které zakladatel akce vytvořil. Pomocí klíče, který je obsažen v kódu iframu aplikace zjistí, o který formulář přesně jde. Aplikace se přesměruje na FormQuestionAnswer.aspx. Na této stránce probíhá vygenerování dotazníku, který uživatel vyplňuje a pak následné uložení výsledků.

Identifikace formuláře, který se má vygenerovat, se provádí pomocí hashkey. Tento klíč je zaznamenán v kódu Iframu. Aplikace pomocí tohoto klíče přesně určí, které otázky patří do jakého formuláře. Načítání dat probíhá ve dvou fázích. V první fázi se pomocí klíče zjistí, jaké otázky se mají načíst z tabulky Query Table. Podle typu otázky, která je uložena v proměnné QueryTypeTable_ID, aplikace zjišťuje, jestli existuje výběr možností, na které bude uživatel odpovídat. Jednotlivé možnosti jsou uloženy v tabulce Answers

[1] mythesis.blob.core.windows.net/calendars/10085T1h3uD9aJwDB400EN0ohfhj.ics
[2] http://mythesis.cloudapp.net/Login.aspx?hashkey=10085T1h3uD9aJwDB400EN0ohfhj



Obrázek 11: Schéma QueryTable



Obrázek 12: Schéma UserAnswerTable

Table. Když jsou všechna data nahrána z databáze, zavolá se metoda AddControl() a umístí se na stránku. Takto vygenerovaná stránka je připravena pro vyplnění uživatele.

Poté co uživatel odpoví na jednotlivé otázky, projdou se všechny Question Controls a uživatelovy odpovědi se zaznamenají. Odpovědi jsou zaznamenány v tabulce User Answer Table. Zadavatel události má tak přehled o tom, kdo a jak odpověděl. Poté co proběhne úspěšné vyplnění dotazníku je uživatel přihlášen k události. Pro jeho vlastní účely je vygenerován kalendář, který si může stáhnout z adresy uvedené na stránce. Ke kalendáři je připojen i odkaz na návrat jak jej importovat v aplikaci Hotmail.

K zaznamenávání odpovědí jednotlivých uživatelů slouží 3 tabulky (obrázek 11). Pomocí těchto tabulek se uloží jednotlivé odpovědi uživatelů a přiřadí se konkrétní otázce.

- **FormTable** obsahuje informace o jednotlivých formulářích, jako například datum a čas konání, název akce.
- **QueryTable** definuje otázky k jednotlivým formulářům. Jsou zde uloženy informace o typu otázky a její konkrétní znění.
- **AnswerTable** ukládá možnosti odpovědí jednotlivých otázek.

3.2.2.3 Vytváření otázek Otázky se tvoří na stránce FormCreateQuestion.aspx. Tuto stránku tvoří jeden control zvaný Form Control. Tento control zobrazuje rozhraní, pomocí kterého uživatel definuje typ otázek a zadává název otázky a případný další popis. Volbu typu otázky si zvolíme v drop down listu. Máme 5 typů otázek. Klasický textbox, kde uživatel napíše libovolnou odpověď, nebo pro delší odpovědi paragraph text. Pokud potřebujeme uživateli zadat nějaké možnosti, máme na výběr 3 možnosti. Buďto možnost více odpovědí v podobě check boxu nebo jen jednu odpověď v podobě drop down listu a pole radiobuttonů. Control se ovládá pomocí 3 tlačítek. Máme možnost přidávat nebo odebírat možnosti dané otázky a pak otázku celou vygenerovat.

3.2.3 Generování objektů

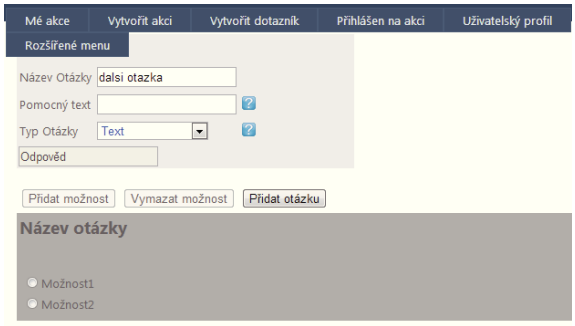
Tvorba jednotlivých objektů se provádí v metodě CreateChildControls. Generuje se zde lokalizace pro všechny objekty, vytvářejí se zde event události. Tato metoda vygeneruje nejprve pevnou kostru rozhraní, které se zobrazí na stránce. Až po kompletním sestavení kostry se začnou generovat objekty na základě toho, jaká byla poslední akce uživatele. Pro změnu typů otázek je důležitá metoda downList_TextChanged. Tato metoda poslouchá na události Text Changed a ukládá do session hodnotu změny. Po vytvoření pevné kostry controlu si metoda Create Child Control zavolá metodu AddItem, ve které se předává parametr právě hodnoty down listu, která je uložena ve view state. Metoda pak podle typu otázky vygeneruje patřičné objekty. Pokud má otázka více možností, počet těchto možností je taktéž uložen ve viewstate.

3.2.3.1 Ovládání kontrolu Uživatel si může pomocí ovládacích prvků přizpůsobovat generování jednotlivých otázek podle svého uvážení. Má na výběr ze 3 možností jak si kontrol přizpůsobit.

- **Změna typu otázky** se provádí pomocí výše zmíněného downListu, lze nastavovat různé druhy otázek.
- **Mazání jednotlivých možností** se provádi pomocí tlačítka btnDeleteItem. Tento objekt má nastavenu vlastnost UseSubmitBehavior na false. Takto nastavené chování nám umožňuje odchytit toto tlačítko v requestu stránky. V metodě AddItem se pak toto tlačítko zachytí a aplikace vygeneruje o jeden prvek méně.
- **Přidávání možností** vykonává tlačítko btnAddItem, jehož událost OnClick poslouchá metoda kontrolu btnItem_Click. Pomocí této metody se zjistí hodnota downListu pomocí viewState a podle hodnoty se přidá ten správný typ možnosti.

3.2.4 Generování otázek

Jestliže je otázka vytvořena, tlačítkem btnAddQuestion ji můžeme vygenerovat. U tohoto tlačítka poslouchá událost metoda kontrolu btn_Click. Pomocí této metody se sesbírají



Obrázek 13: Schéma UserAnswerTable

všechny informace potřebné k vygenerování otázky a umístění na stránku. Počet jednotlivých možností je uložen ve viewstate a metodou FindControl si můžu najít jednotlivé možnosti a zjistit jejich hodnotu. Poté co budou sesbírána všechna data do view state se uloží několik parametrů, které budou poté využívány ke generování při znovunačtení stránky. Jeden z parametrů je celkový počet otázek, které se mají generovat. Druhý parametr je objekt, který sesbíral informace o tom jak má daná otázka vypadat a do view state je uložen pod svým jedinečným identifikačním kódem.

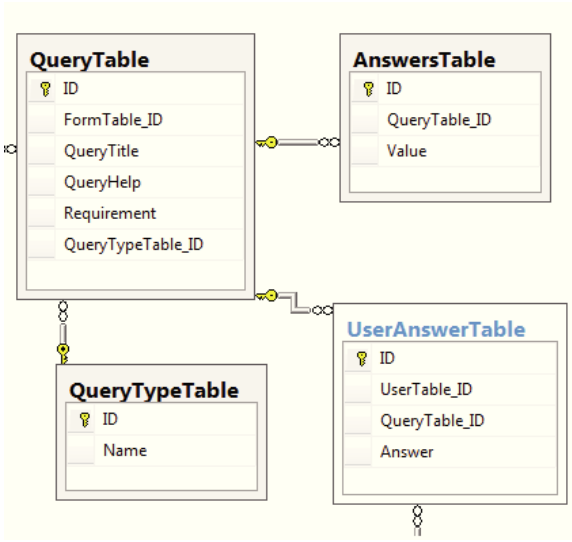
3.2.4.1 Question Control Tento control reprezentuje jednotlivé otázky vytvořené uživatelem, které uživatel generuje na stránku. Každý control se vygeneruje tak, jak uživatel zvolil. Control má jeden aktivní prvek a tím je btnDeleteQuestion, pomocí něhož můžeme jednotlivé controly mazat. Tento prvek má nastaveno UseSubmitBehavior na false. Tento prvek se pak odchytává přímo na stránce v metodě AddControl.

Tato metoda vypisuje jednotlivé Question controly na stránku. Díky metodě btn_Click controlu FormControl jsou veškerá potřebná data uložena ve view state. Pomocí těchto dat se vygenerují otázky na stránku. Na začátku této metody je kontrola, zda v některém z Question control nebylo aktivováno tlačítko pro smazání Question controlu. Pokud ano, pomocí identifikačního čísla se takový control vyhledá a smaže jak z view state, tak případně z databáze.

3.2.5 Ukládání otázek

Pokud máme tvorbu otázek dokončenou, je potřeba je uložit. Tyto otázky jsou uloženy v tabulce Query Table. Zaznamenává se zde znění otázky a případný vysvětlující text. Důležitým atributem je typ otázky. Pomocí tohoto atributu program určí, jestli se jedná o otázku, kde se odpovídá na předem definované možnosti a zda je možno zaznamenat více odpovědí, nebo uživatel může opovědět libovolně do textového pole.

```
public class QueryTable
{

    public string  QueryTitle { get; set; }
```



Obrázek 14: Schéma ukládání otázek

```
        public string  QueryHelp { get; set;  }
        public bool Requirement { get; set;  }
        public int  ID  {  get;  set;  }
        public int  QueryTypeTableID { get; set; }
        public int  FormTableID { get; set;  }


        public QueryTable()
        {
            QueryTitle = "";
            QueryHelp = "";
            Requirement = false;
            ID = -1;
            FormTableID = -1;
            QueryTypeTableID = -1;
        }
    }
```

Výpis 2: Třída QueryTable

Pokud má uživatel odpovídat na otázku s více možnostmi, je nutné zaznamenávat jejich znění. Tyto možnosti se ukládají do tabulky Answers Table. Do této tabulky se ukládá identifikační číslo otázky, na kterou bude uživatel odpovídat a znění jednotlivých možností. Jednotlivé odpovědi přihlašovaných uživatelů se ukládají do tabulky User Answer Table. V této tabulce se zaznamenává identifikační číslo uživatele, který odpovídá na otázku a identifikační číslo dané otázky a samozřejmě odpověď.

3.3 Práce s daty

Pro ukládání dat používá aplikace 3 druhy úložišť. SQL Azure, Azure storage, a Local storage.

3.3.1 SQL Azure

Většina dat, jako jsou údaje o uživatelích, záznamy o akcích či dotaznících a odpovědi uživatelů na otázky, jsou uloženy pomocí SQL Azure. Nad takovýmito daty je nutné provádět operace typu join, různě je třídit a podobně. Pro tyto operace je SQL Azure ideální volba, protože takovéto akce podporuje. Další výhodou tohoto řešení je minimální zásah do kódu již vytvořené stránky. V našem případě stačilo pouze upravit připojovací řetězec, který používáme při SQL dotazech.

```
<connectionStrings>
    <add name="ConnectionString"
    connectionString="Data Source=la0t57h4l7.database.windows.net;Initial Catalog=Database;
          User ID=login;Password=pass"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```

Výpis 3: Příklad definice připojovacího řetezce

Řetězec pro připojení do databáze obsahuje kromě svého názvu také adresu cloudového úložiště. Initial katalog pak označuje konkrétní databázi, na kterou se chceme připojit a nakonec přihlašovací údaje, které jsou používány pro přístup do databáze. Takto vytvořený řetězec se pak používá u metod, které s databází nějakým způsobem komunikuji.

3.3.2 Využití blobu

Blob je nejjednodušší způsob jak uložit nějaká nestrukturovaná data. Jeho největší výhodou, kterou v naší aplikaci potřebujeme, je možnost použití REST. Tedy možnost přistoupit na blob přes http požadavek. Přiklad takovéto adresy může vypadat takto.

- mythesis.blob.core.windows.net/calendars/10085T1h3uD9aJwDB400EN0ohfhj.ics

První část http dotazu mojediplomka.blob.core.windows.net je adresa úložiště, na kterém aplikace běží. Prostřední část Calendars představuje kontejner, ve kterém jsou sdruženy jednotlivé bloby nesoucí informace. Poslední část URL adresy je identifikační číslo kalendáře, pomocí něhož se dostaneme na námi určený soubor. Díky tomuto snadnému řešení je Icalendar kdykoliv k dispozici pro možné aktualizace.

3.3.3 Práce s bloby

S bloby je poněkud složitější práce než s SQL Azure. Tak jako klasické SQL Azure i tady je potřeba nadefinovat připojovací řetězec, pomocí něhož určíme, na který server se mají

data ukládat. Připojovací řetězce pro práci s Azure úložištěm definujeme přes uživatelské rozhraní pomocí něhož nadefinujeme jméno řetězce, název účtu, na kterém úložiště běží a přístupový klíč, který je potřebný pro ověření, zda má uživatel práva na to, aby mohl s daty jakkoliv manipulovat. Takto námi vytvořený řetězec se pak zobrazí v konfiguračním souboru cloudové role s názvem ServiceConfiguration.cscfg

```
<Setting name="BlobConnString" value="DefaultEndpointsProtocol=https;AccountName=
    mojediplomka;AccountKey=myAccKey=" />
```

Výpis 4: Příklad definice připojovacího řetezce

Pro každý konfigurační řetězec se v souboru vytvoří sekce, kde je popsána jeho definice. Na příkladu takového řetězce můžeme vidět sekci Name v ukázce zdrojového kódu připojovacího řetezce. V této sekci je název připojovacího řetězce, na který se pak budeme odkazovat, přímo v aplikaci. Dále nastavení protokolu pro přenos. Máme na výběr ze dvou variant. Buď HTTP, nebo HTTPS. Sekce Name definuje název úložiště a Account Key pak ověřovací klíč pro přístup do takovéhoto úložiště. Přístupový klíč je vyžadován pouze k manipulaci s daty. V případě pouhého čtení anonymními uživateli klíč vyžadován není.

3.3.4 Ukládání blobu

Jako první věc, kterou při práci s bloby musíme udělat, je vytvoření třídy CloudStorageAccount. Tato třída reprezentuje cloudové úložiště, se kterým budeme chtít pracovat. Referenci na dané úložiště získáme pomocí metody Get Configuration Setting Value. Tato statická metoda je součástí třídy RoleEnviroment. Pomocí této třídy získáváme data z konfiguračního souboru, kde jsou uložena nastavení jednotlivých úložišť nebo endpointu. Metodě GetConfigurationSettingValue jako parametr předáváme název připojovacího řetězce, kde je nakonfigurovaná cesta k úložišti a přihlašovací údaje nutné pro práci s daty. Na základě získaných dat o Azure úložišti můžeme vytvořit třídu CloudBlobClient. Tato třída umožní komunikaci s blob storage. Komunikace se naváže díky metodě CreateCloudBlobClient. Poslední část je získání reference na kontejner, ve kterém jsou jednotlivé bloby uloženy. Toho docílíme díky metodě GetContainerReference, které v parametru metody předám název kontejneru. Takto získanou referenci přidělím třídě CloudBlobContainer. Nyní už mám plný přístup ke všem blobům v daném kontejneru.

```
//Získáni Azure účtu pomoci připojovacího řetězce.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(RoleEnvironment.
    GetConfigurationSettingValue("BlobConnString"));

//Navázání komunikace s blob storage
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

//Získání reference na konkrétní kontejner blobu.
CloudBlobContainer container = blobClient.GetContainerReference("calendars");
container.CreateIfNotExist();
```

Výpis 5: Příklad získání reference na kontejner

Jak ukazuje příklad, nejprve se pomocí řetězce připojím na účet, kde jsou data, která hledám. Poté se vytvoří instance blob klienta, která se připojí na výše definovaný účet. Poslední řádek je pak vytvoření kontejneru, do něhož budeme ukládat data. V našem případě se data nahrávají z lokálního úložiště nazvané Calendars, které je uloženo na virtuálním počítači kde aplikace běží.

3.3.5 Přístupová práva

U blobů a celého kontejneru, ve kterém jsou uložena, můžeme nastavovat práva přístupu k informacím, která obsahují. Nastavení práv se provádí pomocí metody SetPermissions.

```
container.SetPermissions(new BlobContainerPermissions { PublicAccess =
    BlobContainerPublicAccessType.Blob });
```

Výpis 6: Nastavení práv blobu

Této metodě se předá argument BlobContainerPublicAccessType. Tímto argumentem definujeme přístupová práva k danému blobu. Tato práva nabývají tří hodnot:

- **Nastavení blob**, které používáme v naší aplikaci. Umožňuje anonymním uživatelům číst data obsažena v blobu a také číst metadata daného blobu. Nemohou však číst metadata kontejneru, kde je blob uložen.
- **Nastavení Cantainer** umožňuje číst data v blobu a jeho metadata. Může číst i metadata kontejneru.
- **Nastavení Off**. Zakazuje anonymní přístup. Jediný kdo má přístup do kontejneru je vlastník účtu.

3.3.6 Lokální úložiště

Lokální úložiště je ideální cesta kam ukládat data dočasného charakteru jako jsou různé mezivýsledky, nebo data, která se budou dále zpracovávat. Lokální úložiště je nutné nadefinovat v uživatelském rozhraní cloudové role v sekci Local Storage. V této sekci zadáváme kromě názvu také velikost, jakou bude úložiště mít. Stejně jako řetězec pro připojení pomocí blobu, tak i tato konfigurace se objeví v konfiguračním souboru ServiceConfiguration.cscfg. V naší aplikaci se lokální úložiště užívá při tvorbě kalendářů na vytvářenou událost. V průběhu zápisu se data pro kalendář zapisují na lokální úložiště. Jakmile jsou všechna data seskupena, pomocí blobu si data vytáhneme a uložíme do trvalého úložiště na cloud. K lokálnímu úložišti nemůžeme přistupovat jako k blobům, tedy přes url adresy. K úložišti má přístup jen aplikace běžící na daném virtuálním stroji. Referenci na lokální úložiště reprezentuje třída zvaná Local Resource. Tuto třídu spojíme s naším lokálním úložištěm pomocí statické metody Get Local Resource, které v parametru

Počet uložení	1	10	100	1000
Čas (s) blob úložiště	11,50	11,53	16,90	79,00
Čas (s) SQL úložiště	11,67	11,60	16,11	61,25
Čas (s) lokálního úložiště	11,11	11,38	11,45	12,60

Tabulka 1: Naměřené hodnoty úložišť

předáme název nakonfigurovaného úložiště. Tato statická metoda je součástí třídy pro získávání nakonfigurovaných dat zvanou Role Enviroment. Jakmile jsme získali referenci na potřebné úložiště, zbývá nám už jen zjistit cestu. Cesta k úložišti je uložena v proměnné třídy Local Resource Rooth Path.

```
// Získání reference na potřebné úložiště
LocalResource localResource = RoleEnvironment.GetLocalResource("Calendars");

// Nastavení cesty ke konkrétnímu souboru v lokálním úložišti. Jméno souboru je jedinečný
    identifikační znak uložený v session.
string filePath = Path.Combine(localResource.RootPath, Convert.ToString(Session["hashkey"]) + "
    .ics");
```

Výpis 7: Příklad získání souboru z lokálního úložiště

3.3.7 Porovnání úložišť

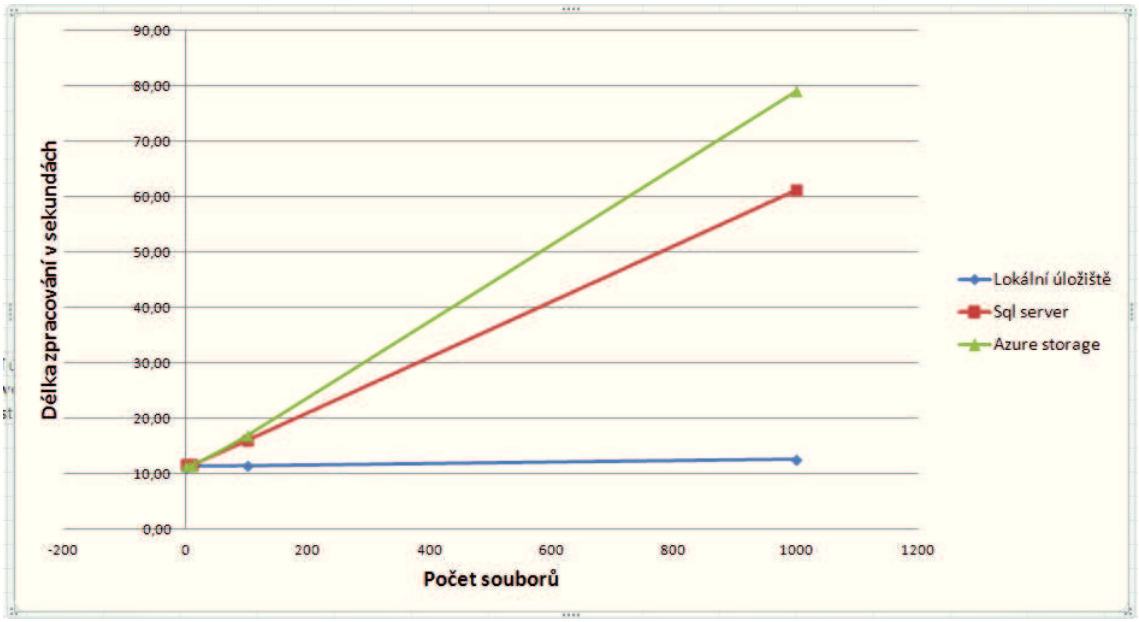
Každé úložiště má své specifické výhody. SQL Azure disponuje všemi funkcemi klasického SQL serveru a je proto vhodný pro data, u kterých potřebujeme využívat různé druhy spojování tabulek, třídění, nebo různé typy výpisů. Bloby mají zase výhodu možnosti pracovat s daty přes http požadavek. Jak jsou ale jednotlivá úložiště efektivní pro práci s daty.

Testovány byly 3 typy úložišť:

- **Lokální úložiště**
- **Úložiště blobu**
- **SQL azure**

K testu byl použit obrázkový soubor o velikosti 284 kB. Tento soubor byl opakovaně ukládán do jednotlivých úložišť. Tento postup byl zvolen z důvodu absence dostatečného množství dat v úložištích aplikace, nad kterými by se mohl tento test provádět. Testování Azure úložiště se provádělo prostřednictvím blobu.

Jak ukazují jednotlivé tabulky jedna až tři, soubor byl ukládán do úložiště ve čtyřech fázích. V první fázi byl uložen pouze jednou. Postupně počet souborů narůstal. Čas, uváděn v sekundách, je doba od začátku požadavku po jeho ukončení.



Obrázek 15: Graf doby zpracování požadavku

3.3.7.1 Výsledky testování V tabulce 1 jsou zobrazeny časy jednotlivých ukládání souboru pomocí blobu. Časy tohoto úložiště byly nejpomalejší a nejstrměji rostly. V tabulce 2 jsou zapsány pokusy prostřednictvím SQL Azure. Tento typ byl jen o něco pomalejší než ukládání přes bloby, ale nárust byl o něco mírnější než u blobu. Nejlépe ze všech testovaných úložišť dopadlo lokální úložiště, které je zobrazeno v tabulce 3. Čas s narůstajícím počtem souborů rostl jen nepatrně. Tento typ úložiště je tedy vhodný pro zpracovávání velkého množství dat. Jeho značnou nevýhodou je pak izolovanost od jiných lokálních úložišť. Ostatní úložiště měla podobný nárust doby zpracování s přibývajícím množstvím souborů jak je možné vidět na obrázku 15, který znázorňuje graf.

4 Unit testy

Podstata unit testování spočívá v izolaci malé části programu od zbytku testovaného kódu. U takto izolované části můžeme ověřit, jestli se chová přesně tak jak očekáváme. Cílem takovéhoto testování je oddělit jednotlivé části kódu a zjistit, zdali jsou správně napsány či nikoliv. Unit test poskytuje striktní popis toho, co daný kód a za jakých podmínek musí splňovat. Z tohoto chování plyne několik výhod.

4.1 Výhody

- **Usnadňování změn.** Unit testy umožňují vývojáři po delší době znovu prověřit kód a ujistit se, že ta daná část kódu pořád funguje stejně kvalitně. Smyslem je napsat testy pro všechny případy funkcí a metod. Takže pokud nějaká změna způsobí špatné chování systému, může být jednoduše nalezena a odstraněna. Snadná dostupnost unit testu poskytuje programátorovi možnost rychle a jednoduše zjistit jestli jeho kód funguje pořád správně. Unit testy programátorovi poskytují neustálou zpětnou vazbu toho, jakým způsobem by měly dané metody fungovat, i když budou v kódu prováděny neustálé změny.
- **Zjednodušení integrace.** Unit testy poskytují jakousi živou dokumentaci. Programátor, který se chce dozvědět jak daná metoda nebo část kódu funguje, se může podívat na unit test a získá základní přehled o tom jak daná metoda funguje. V některých případech unit testy ztělesňují chování, které je klíčové pro úspěch dané části kódu. Tyto charakteristiky chování můžou indikovat úspěch či neúspěch. Unit testy může být totiž zachyceno i nežádoucí negativní chováni, které by mohlo mít vliv na ostatní části kódu.

4.2 Nevýhody

U unit testu nemůžeme očekávat, že zachytí každý error v programu. Testy testují jen funkčnost jednotek a nesledují různé další integrace, ani nezachytí nějaké rozsáhlé systémové chyby. Unit testování by mělo byt součásti nějakého celku, který se zabývá testováním softwaru. Dalším možným problémem unit testu je testování samotné. Například na každý boolovsky problém, který nabývá hodnot true nebo false potřebuju minimálně dva unit testy. Jeden na odpověď true, druhý na odpověď false. Ve výsledku to znamená, že na každý řádek kódu napsaného programátorem většinou potřebujeme 3 až 5 řádků unit testového kódu. To obvykle zabere nějaký čas a investice se ne vždy může vyplatit.

4.3 Testování ve Visual studiu

Do prostředí Visual Studia byl integrován Framework pro unit testování. Tento Framework umožňuje generování kódu, spouštění testů přímo v prostředí Visual Studia, nebo spouštění unit testů s daty, které jsou uloženy v databázi. Unit testy můžeme použít na jakýkoliv projekt, stačí pouze přidat k již existující aplikaci nový projekt zvaný Test Projekt, který je umístěn v sekci C#.

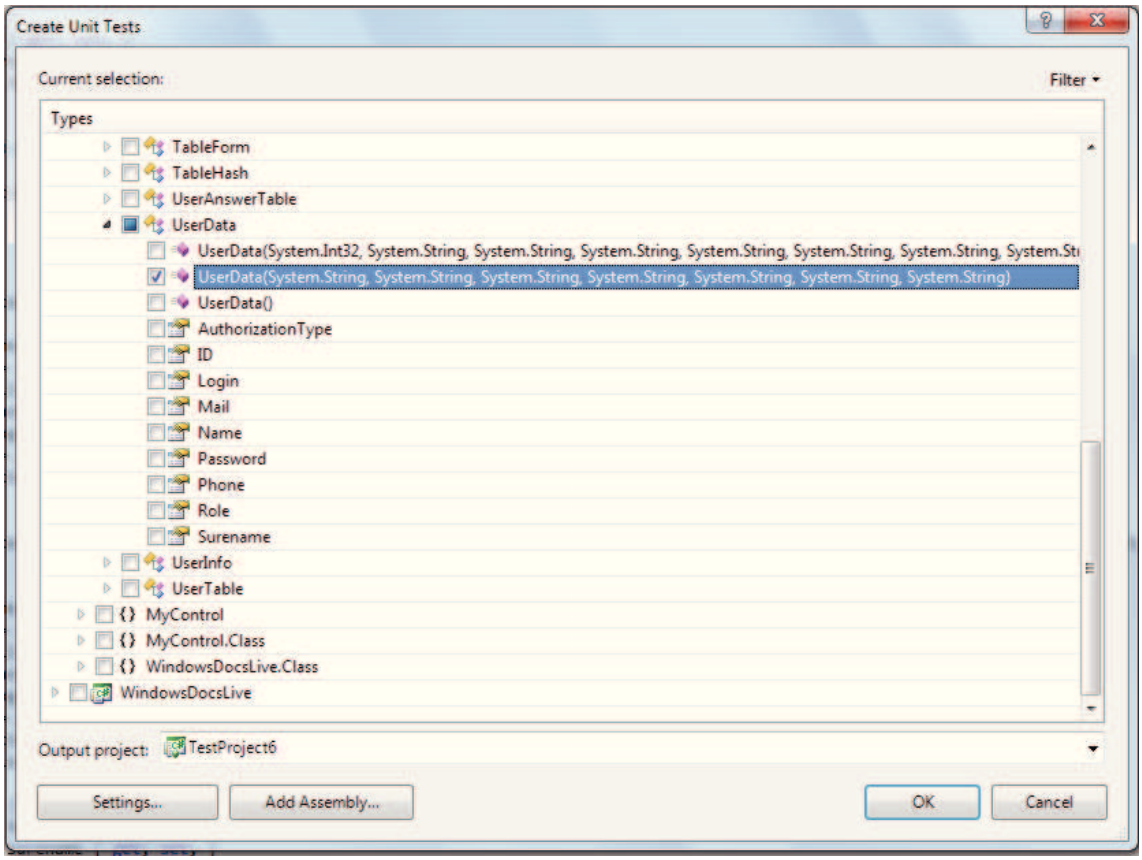
4.3.1 Vytvoření testu

Mějme jednoduchou třídu, která reprezentuje tabulku v databázi zvanou User Table. Tato třída si v konstruktoru předá informaci o přihlašovacích údajích uživatele, jako je přihlašovací jméno a heslo, ale také uživatelovy osobní údaje.

```
public class UserData
{
    public UserData()
    {

    }

    public UserData(string name, string surename, string mail, string phone, string login, string
        password, string authoriyationtype)
    {
        Name = name;
        Surename = surename;
        Mail = mail;
        Phone = phone;
        Login = login;
        Password = password;
        AuthorizationType = authoriyationtype;
    }

    public UserData(int id, string name, string surename, string mail, string phone, string login,
        string password, string authoriyationtype, int role)
    {
        ID = id;
        Name = name;
        Surename = surename;
        Mail = mail;
        Phone = phone;
        Login = login;
        Password = password;
        AuthorizationType = authoriyationtype;
        Role = role;
    }
    public int ID { get; set; }
    public string Name { get; set; }
    public string Surename { get; set; }
    public string Mail { get; set; }
    public string Phone { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public string AuthorizationType { get; set; }
    public int Role { get; set; }
}
```

Výpis 8: Příklad třídy UserData

Pokud chceme vytvořit nad takovýmto kódem Unit test, stačí kliknout na metodu, kterou chceme testovat a z nabídky vybrat Unit test.



Obrázek 16: Menu tvorby UnitTestu

Jak je možné vidět na obrázku 8. Při tvorbě testů máme možnost výběru, pro které proměnné se má testovací kód vygenerovat. Unit test lze také přiřadit k určitému již existujícímu projektu, nebo si vytvořit nový testovací projekt. Testovací projekty nemusí být jen typu C#, ale i typu C++ nebo Visual Basic. Po nastavení všech proměnných a typu projektu se vytvoří třída, které se za jméno testované třídy přidá slovo Test a bude uložena v projektu buďto nově vytvořeném nebo v již existujícím podle toho, jakou možnost jsme zvolili. Následný vygenerovaný kód vypadá v našem případě následovně.

```
namespace TestProject6
{


    /// <summary>
    ///This is a test class for UserDataTest and is intended
    ///to contain all UserDataTest Unit Tests
    ///</summary>
    [TestClass()]
    public class UserDataTest
    {


        private TestContext testContextInstance;

        /// <summary>
        ///Gets or sets the test context which provides
        ///information about and functionality for the current test run.
        ///</summary>
        public TestContext TestContext
        {
            get
            {
                return testContextInstance;
            }
            set
            {
                testContextInstance = value;
            }
        }

        /// <summary>
        ///A test for UserData Constructor
        ///</summary>
        [TestMethod()]
        public void UserDataConstructorTest()
        {
            string name = string.Empty; // TODO: Initialize to an appropriate value
            string surename = string.Empty; // TODO: Initialize to an appropriate value
            string mail = string.Empty; // TODO: Initialize to an appropriate value
            string phone = string.Empty; // TODO: Initialize to an appropriate value
            string login = string.Empty; // TODO: Initialize to an appropriate value
            string password = string.Empty; // TODO: Initialize to an appropriate value
            string authoriyationtype = string.Empty; // TODO: Initialize to an appropriate value
```

```
            UserData target = new UserData(name, surename, mail, phone, login, password,
                authoriyationtype);
            Assert.Inconclusive("TODO: Implement code to verify target");
        }
    }
}
```

Výpis 9: Příklad automaticky vygenerovaného kódu

Ve vygenerovaném kódu jsou nejdůležitější dva atributy. Tím prvním je atribut Test Method. Tento atribut označuje metodu, která se bude testovat. Druhým atributem je Test Class, který označuje třídu obsahující testovací metody. Oba tyto atributy lze nalézt ve jmenném prostoru Microsoft.Visual Studio.Quality Tools.Unit Testing.Framework. Team test používá reflexi k hledání testovacích tříd. Orientuje se podle atributu Test Class. Poté co najde všechny takto označené třídy, začne hledat metody, které jsou označeny atributem Test Method. Testovaná třída UserDataConstructorTest obsahuje kromě vybraných proměnných a reference na testovanou třídu také třídu Assert s atributem Inconclusive. Tento atribut určuje, že vygenerovaná metoda je doposud nefunkční a uživatel musí doplnit kód, který se bude používat k testování. Následný kód může vypadat takto.

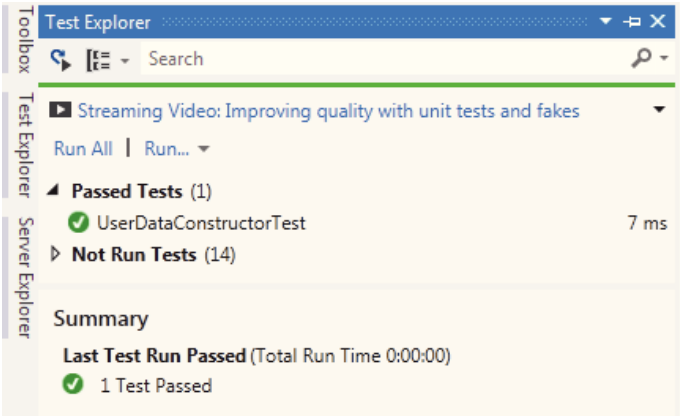
```
[TestMethod()]
        public void UserDataConstructorTest()
        {
            string name = "Marek";
            string surename = "Markovic";
            string mail = "neco@nekde.cz";
            string phone = "785695632";
            string login = "Uzivatel";
            string password = "heslo";
            string authoriyationtype = "FORMS";
            UserData target = new UserData(name, surename, mail, phone, login, password,
                authoriyationtype);
            Assert.AreEqual(name, target.Name);
            Assert.AreEqual(surename, target.Surename);
            Assert.AreEqual(login, target.Login);
            Assert.AreEqual(password, target.Password);
        }
```

Výpis 10: Kód UnitTestu. který je připraven k použití

Assert.Inconclusive jsme přepsali generickou metodou Assert.Are Equal. Pomocí této metody budeme porovnávat přicházející data s daty uvnitř třídy. Tato metoda přijímá dva parametry, které porovnává. Do lokálních proměnných uložíme příkladová data. Stejná data pak pošleme do konstruktoru třídy UserData. Metoda Are Equal pak porovná výsledky a pokud budou stejná, test projde. V opačném případě test selže.

4.3.2 Spouštění testu

Poté co se testovací kód připraví, můžeme začít s testováním. K testování je nutné nastavit vytvořený testovací projekt jako spouštěcí a normálně spustit aplikaci.



Obrázek 17: Výsledek testu

Jak je možné vidět na obrázku 9. V okně test result se zobrazují výsledky jednotlivých testů. Lze si zde prohlédnout i detailní popis proč daný test byl neúspěšný a tím snáze odhalit případný problém. V našem případě test prošel. Testovaný konstruktor tedy vytvořil hodnoty podle našeho očekávání. Testovat nemusíme jen inicializací tříd, ale testovat lze i různé výpočetní metody, u kterých víme, při zadání určitých dat, i výsledek. Tento výsledek pak můžeme ověřovat a máme tak jistotu, že tělo metody funguje pořád správně, i když jsou v kódu prováděny změny.

4.3.3 Testování databázových dat

Prostřednictvím Unit testů lze testovat také data v databázi, při využití metod, které s databázi komunikují. Tímto způsobem tak lze ověřovat přístup a funkci metod, které s databází komunikují.

```
public static void FormCreate(FormTable table)
    {
        using (SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["
            ConnectionString"].ToString()))
        using (SqlCommand cmd = new SqlCommand("INSERT␣INTO␣FormTable␣(Name,
            Description,Date,EndTime,Place,DateToLogin,ActionType,HashKey,Anonymous)␣
            values(@Name,@Description,@Date,@EndTime,@Place,@DateToLogin,
            @ActionType,@HashKey,@anonymous)", con))
        {
            cmd.Parameters.Add("@Name", SqlDbType.NVarChar).Value = table.Name;
            cmd.Parameters.Add("@Description", SqlDbType.Text).Value = table.Description;
            cmd.Parameters.Add("@Date", SqlDbType.SmallDateTime).Value = table.Date;
            cmd.Parameters.Add("@EndTime", SqlDbType.SmallDateTime).Value = table.EndTime;
            cmd.Parameters.Add("@Place", SqlDbType.NVarChar).Value = table.Place;
            cmd.Parameters.Add("@DateToLogin", SqlDbType.SmallDateTime).Value = table.
                DateToLogin;
            cmd.Parameters.Add("@ActionType", SqlDbType.NVarChar).Value = table.ActionType;
            cmd.Parameters.Add("@HashKey", SqlDbType.NVarChar).Value = table.HashKey;
```

```
            cmd.Parameters.Add("@anonymous", SqlDbType.Bit).Value = table.Anonymous;

            con.Open();
            cmd.ExecuteNonQuery();
        }
    }
```

Výpis 11: Přiklad testované metody pro ukládání dat do databáze

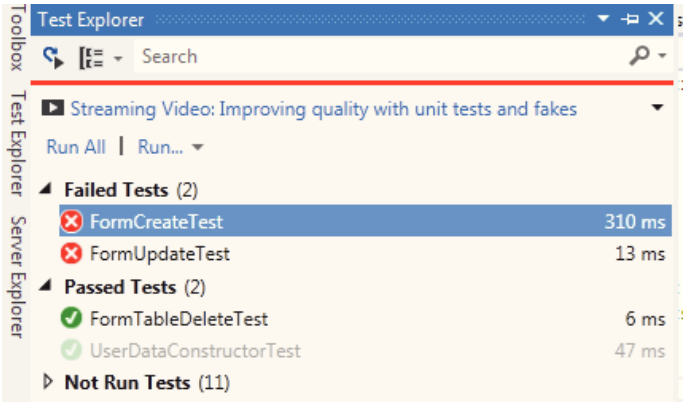
Tato metoda, která slouží k vytváření formulářů se připojí do databáze pomocí připojovacího řetězce, který je uložen v konfuguračním souboru app.config. Tento soubor slouží ukládání konfiguračních informací pro testovací projekt, ve kteerém jsou uloženy testovací třídy. Parametrem metody se předají informace o objektu, který se má ukládat do databáze. Po načteních všech relevantních dat se pomocí SQL příkazu data uloží do databaze. Pro testování takovéto metody jednoduše z nabídky možností vybereme Create Unit Test. Visual studio automaticky vygeneruje soubor, který použijeme k testování.

```
/// <summary>
        ///A test for FormCreate
        ///</summary>
        [TestMethod()]
        public void FormCreateTest1()
        {
            FormTable table = null; // TODO: Initialize to an appropriate value
            MethodsForm.FormCreate(table);
            Assert.Inconclusive("A method that does not return a value cannot be verified.");
        }
```

Výpis 12: Přiklad vygenerované metody pro ukládání dat do databáze

Takto vytvořená testovací metoda obasahuje objekt, který je parametrem testované metody a volání metody samotné. Takto vygenerovaný kód je nutné opět doplnit o data, která budeme používat při testování.

```
/// <summary>
        ///A test for FormCreate
        ///</summary>
        [TestMethod()]
        public void FormCreateTest()
        {
            FormTable table = new FormTable();
            table.ActionType = "Action";
            table.Anonymous = false;
            table.Date = DateTime.Now.AddDays(2);
            table.DateToLogin = DateTime.Now.AddDays(1);
            table.Description = "popis";
            table.EndTime = DateTime.Now.AddDays(3);
            table.Name = "test";
            table.Place = "A303";
            table.HashKey = "H152U5ikjh59liasyert5586qqwe";
            MethodsForm.FormCreate(table);
```



Obrázek 18: Příklad selhání unit testu

```
        FormTable expected = MethodsForm.FormTableSelectOne(MethodsForm.
            GetFormTableID());
        Assert.AreEqual(expected.HashKey, table.HashKey);
    }
```

Výpis 13: Příklad testovací metody pro ukládání dat do databáze

Do objektu FormTable se uložila jednotlivá data a provedlo se pomocí metody FormCreate uložení dat do databáze. Aby bylo možné zjistit jestli uložení proběhlo správně, je nutné data znovu načíst a porovnat s vkládanými daty. Pro porovnání dat se použije opět metoda Asert.AreEqual. V tomto příkladu se použilo porovnání hašovacího klíče. Pokud se oba klíče, jak v databázi tak námi vytvořený budou shodovat, test projde. V případě, že by test selhal, Visual Studio zobrazí ve výsledkovém okně, který z testů neprošel (obrázek 11). Těmito testy byly pokryty všechny metody, které komunikují s databází a ukládají tam důležité informace o uživatelích a jejích akcích. V případě, že by databáze nebo samotné metody přestaly fungovat tak jak očekávám, použitím unit testů mohu snadno přijít na závadu, která se mohla objevit v průběhu vývoje aplikace.

5 ICalendar

I Calendar je program umožňující internetovým uživatelům posílat pozvánky na události ostatním uživatelům internetu. Umožňuje také zasílání emailů nebo sdílení souborů. Příjemci dat z kalendáře mohou jednoduše reagovat na zasílatelovy požadavky. Icalendar byl vyvinut Franken Dawsonem z Lotus Development Corporation a Derikem Steenersonem z Microsoft Corporation. Data Icalendare jsou ve formě prostého textu. Soubory mají příponu .ics. Soubory .ifb obsahuji informace o datu a času. Icalendar byl navrhován, aby byl nezávislý na přenosovém protokolu. Formát Icalendare byl navržen na přenos kalendářových dat, jako jsou například události. Neřeší však, co se s daty bude dít dál. Icalendar je určený k poskytnutí obecného formátu pro výměnu kalendářních a plánovacích informaci pomocí internetu. Zatímco většina funkcí používaná uživateli je široce podporována Icalendarem, některé nadstandardní nástroje mají problémy. Například Icalendář není kompatibilní s ne gregoriánskými kalendáři jako je například lunární kalendář používaný v Izraeli nebo Saudské Arábii.

5.1 Core objekt

V core objektu jsou uloženy kalendářní a plánovací informace. První a poslední řádek core objektu jsou pevně dány. První řádek musí obsahovat "BEGIN:VCALENDAR" na posledním řádku pak musí být ukončení v podobě "END:VCALENDAR" výrazy mezi těmito příkazy se nazývají tělo objektu (iCalbody).

5.2 Tělo kalendáře

Tělo objektu se skládá ze seznamu vlastnosti kalendáře a alespoň jedné komponenty. Tyto vlastnosti se aplikují na celý kalendář. Komponentou kalendáře je několik kalendářních vlastností, které vytvarují schéma (design) kalendáře. Například komponenta kalendáře může představovat událost, úkol, nebo alarm.

5.2.1 Events

Events (VEVENT) popisuje událost, která je naplánována na nějaký časový úsek. Časový úsek je ohraničen proměnnými DTSTART a DTEND. DTSTART reprezentuje začátek události. DTEND naopak představuje datum a čas kdy událost končí. V Events může být také vložena proměnná VALARM, která na začátek události upozorní. V případě opakujících se událostí jako jsou narozeniny, výročí a podobně se proměnná DTEND neuvádí. Event může být použit i bez konkrétního času, jako jsou výročí nebo denní povinnosti. Příklad Eventu níže.

```
BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:2.0
BEGIN:VEVENT
DTSTART:20120329T150000Z
```

```
DTEND:20120330T160000Z
LOCATION:C523
SUMMARY:testovaci 1
UID:iCalendar634696308933230383
DESCRIPTION:
END:VEVENT
END:VCALENDAREVENT
```

Výpis 14: Příklad těla ICalendáře

5.2.2 To-Do

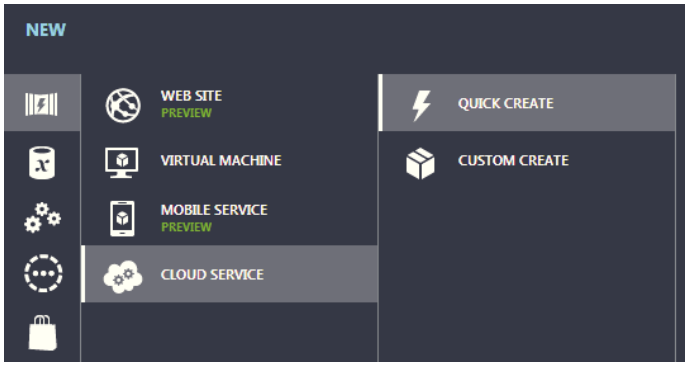
Sekce To-Do(VTODO) definuje něco co je třeba vykonat.

6 Produkční nasazení

Pro to, aby aplikace byla spustitelná ve Windows Azure, je nutné provést několik kroků, kterými připravíme apliakci k nasazení na cloud. Také je potřeba připravit prostředí, ve kterém aplikace poběží.

6.1 Založení úložiště

K používání úložiště dat a aplikací je nutné mít vytvořený účet na Live, který je spjatý s Azure úložištěm. Po přihlášení na windows.azure.com se zpřístupní prostředí, ve kterém lze ovládat jak datové úložiště, tak správu aplikace, firewallu a mnoho dalšího. Jako první věc s novým přihlášením je třeba vytvořit cloudovou službu. Tuto službu bude využívat aplikace pro komunikaci s Windows Azure. Novou službu vytvoříme kliknutím na dolní tlačítko New a vybereme Cloud service a poté Quic create (obrázek 19).



Obrázek 19: menu pro vytvoření nové služby

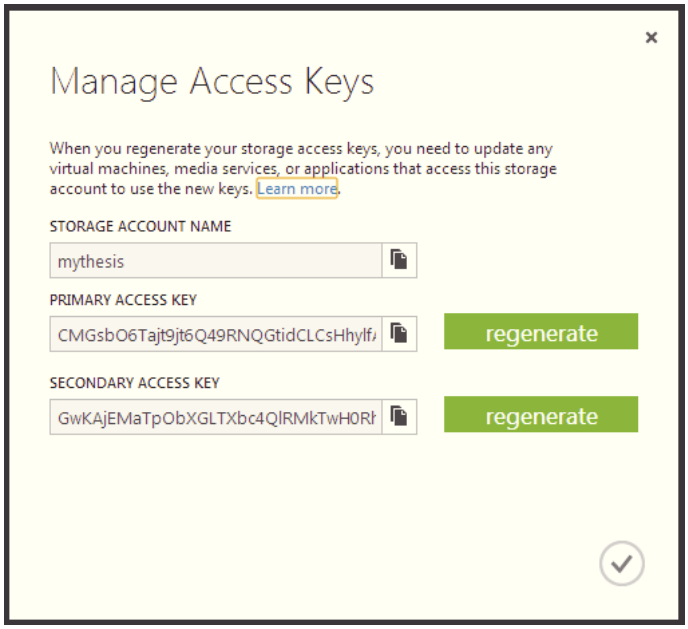
Při zakládaní účtu máme možnost si vybrat destinaci datového centra a musíme si zvolit url adresu, jejímž prostřednictvím bude aplikace přístupná. Po vytvoření účtu v okně vlastností máme seznam nejrůznějších informací, jako jsou url adresy blobu nebo tabulek. Můžeme zde vidět region a jméno, které jsme zadávali při vytváření.

Pro možnost ukládat data jinam než do tabulek musíme vytvořit úložiště pro bloby. V menu klikneme opět na New a vybereme Storage a Quick create. Opět se zadá lokace datového centra a podobu url adresy přes kterou budeme přistupovat k blobům.

Nejdůležitější položkou je Primary a Secondary Access key(obrázek 20). Tyto přístupové klíče používá azure k ověření aplikací, které chtějí s úložištěm komunikovat. Secondary Access key je záložní klíč, který se používá v případech kdy Primary Access key je nějakým způsobem nepoužitelný.

6.2 Nasazení aplikace

Pro to, aby aplikace mohla být nasazena na Azure, je nutné v ní provést několik úprav. K již existujícímu projektu je nutné přidat cloud projekt. Po přidání projektu v záložce role



Obrázek 20: Přístupové klíče k úložišti

vybereme možnost přidání nové a vybereme původní projekt, který chceme nasadit na cloud. Dále je zapotřebí přidat reference na knihovny Azure, které budeme potřebovat. Jsou to:

- Microsoft.WindowsAzure.Diagnostics
- Microsoft.WindowsAzure.ServiceRuntime
- Microsoft.WindowsAzure.Diagnostics.StorageClient

V souboru Global.asax najdeme metodu Application_Start. Do této metody vložíme službu, která nám umožní volat konfigurační parametry, které jsou definovány pro úložiště. Pro inicializaci této služby se použije Cloud Storage AccountṠet Configuration Settings Publisher, díky této třídě můžeme volat jednotlivé parametry pomocí třídy Cloud Storage AccountḞrom Configuration Setting. Tato třída už vyžaduje jen název parametru, který chceme.

```
Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher((configName,
    configSetter) =>
        {
            configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
        });
```

Výpis 15: Příklad nastavení služby

Dalším krokem je nastavení samotného úložiště. V souboru ServiceConfiguration.cscfg najdeme větev ConfigurationSettings a přidáme definici úložiště.

```
<Setting name="DataConnectionString"
                value="DefaultEndpointsProtocol=https;AccountName=name;AccountKey=acesskey
                    " />
```

Výpis 16: Příklad nastavení služby

Parametr Name reprezentuje název nastavení. Do položky AccountName vkládáme název naší servisní služby a do AccountKey vložíme buďto primární nebo sekundární klíč.

Nastaveni aplikace je dokončeno. Nyní stačí aplikaci zabalit do balíčku a přes webové rozhraní odeslat na Azure. Nahrávání může trvat i několik desítek minut.

6.3 Připojení databáze

Pomocí portálu windowsazure.com můžeme vytvářet, odstraňovat, nebo jinak upravovat databáze. Při vytváření nového databázového serveru opět vybíráme datové centrum, kde chceme data ukládat a také zakládáme administrátorský účet, který se dá použít při přihlašování přes SQL server, nebo přes rozhraní, pomocí něhož můžeme tabulky různě upravovat. Po vytvoření serveru je potřeba nastavit Firewallova pravidla. Je to seznam IP adres, ze kterých bude povolen přístup do databází. Výchozí nastavení zabraňuje jakémukoliv přístupu, tedy je nutné tato pravidla nastavit. Po vytvoření serveru můžeme začít vytvářet jednotlivé databáze. Při vytváření databáze můžeme nastavit kromě názvu také velikost. Pro vytváření uložených procedur nebo tabulek vybereme v panelu nástrojů nabídku manage. Dostaneme se tak do rozhraní, kde budeme schopni upravovat jak tabulky, tak i její jednotlivé řádky.

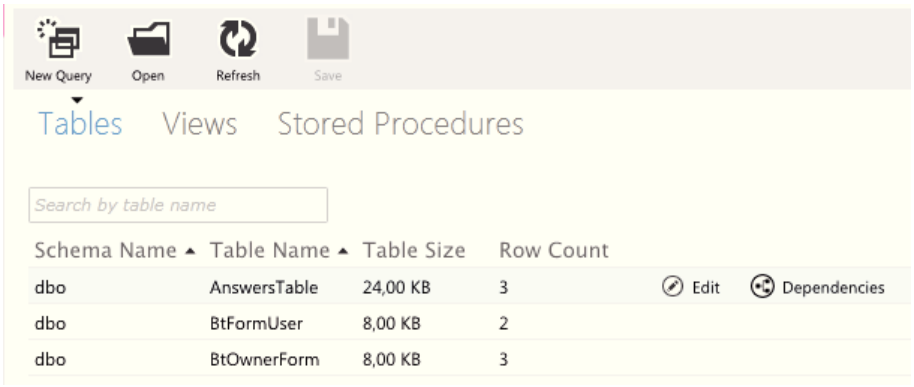
6.3.1 Rozhraní pro práci s databází

V tomto rozhraní máme plný přístup ke všem datům a tabulkám. Tabulky můžeme vytvářet, upravovat, nebo mazat. Máme přehled o využitém místě databáze. Výhodou tohoto rozhraní je, že data lze upravovat, i když je tabulka zaplněna daty. Můžeme je buďto přepisovat, vkládat nebo jim měnit datové typy, což v klasickém SQL serveru možné není. Data lze i testovat pomocí dotazu.

6.3.2 Nastavení pro přípojení

Propojeni aplikace a databáze se provede jednoduchým nadefinováním připojovacího řetězce v konfiguračním souboru Web.config. V sekci connectionstring uvedeme jméno řetězce a následný propojovací řetězec. Do údaje ID vyplňujeme login účtu pro danou databázi. Password je pak příslušné heslo od daného účtu.

```
<connectionStrings>
```



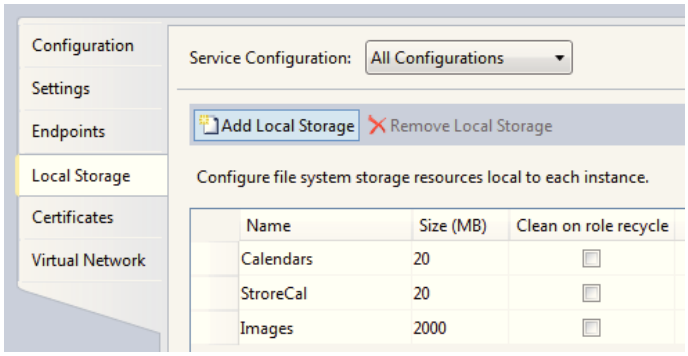
Obrázek 21: Rozhraní pro úpravu tabulek

```
<add name="ConnectionString" connectionString="Data Source=la0t57h4l7.database.windows.
    net;Initial Catalog=Database;User ID=login;Password=pass"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Výpis 17: Příklad nastavení připojovacího řetězce

Po nastavení přístupu do SQL Azure můžeme využívat veškerých výhod, které nám klasický SQL server poskytuje. Žádná další úprava není potřeba. Dotazovací řetězce vytváříme úplně stejně jako kdyby databáze běžela na klasickém SQL serveru.

6.3.2.1 Lokální úložiště Při používání lokálního úložiště je nastavení a používání o něco složitější. Je třeba lokální úložiště nejen nadefinovat, ale je nutný i zásah do kódu aplikace. Úložiště se nadefinuje v konfiguraci cloudového projektu (obrazek 21). V sekci



Obrázek 22: Menu lokálního úložiště

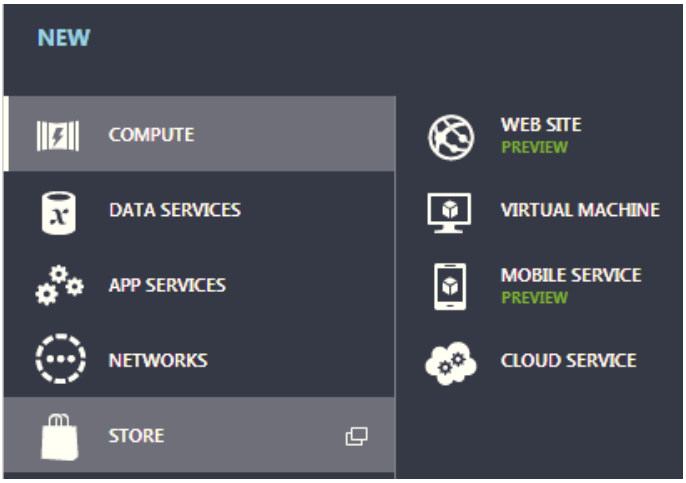
LocalStorage nadefinujeme úložiště. Zadáme jméno a velikost. V našem případě jsou nadefinovány 3 lokální úložiště. Každému úložišti se přiřadí velikost v MB. Atribut Clean on role recycle udává, jestli data v lokálním úložišti přežijí životní cyklus dané role.

Implicitně je tato hodnota na true, což znamená, že data budou při znovu vytvořeni role vymazána.

6.4 Aktivace mailového klienta

Zasílání emailů prostřednictvím cloudu má svá mnohá omezení, jako například množství emailových zpráv, které můžeme zaslat mesíčně, nebo přístup k některým službám usnadňující nám správu emailového účtu. Tato omezení mají několik důvodů. Ten nejzávažnější je ten, že pokud by cloudová centra byla zneužíta k posílání nežadoucích emailových zpráv, mohla by se tato centra dostat na černou listinu takto poškozovaných serverů a byli by poškození všichni zákazníci, kteří dané centrum využívají.

Služeb, které zajišťují zasílání emailových zpráv jako třetí strana je mnoho. V aplikaci je použita služba SendGrid, kterou lze nastavit přímo v prostředí Windows Azure. V dolní nabídce vybereme možnost Store (obrázek 24).



Obrázek 23: Menu pro výběr služeb Azure

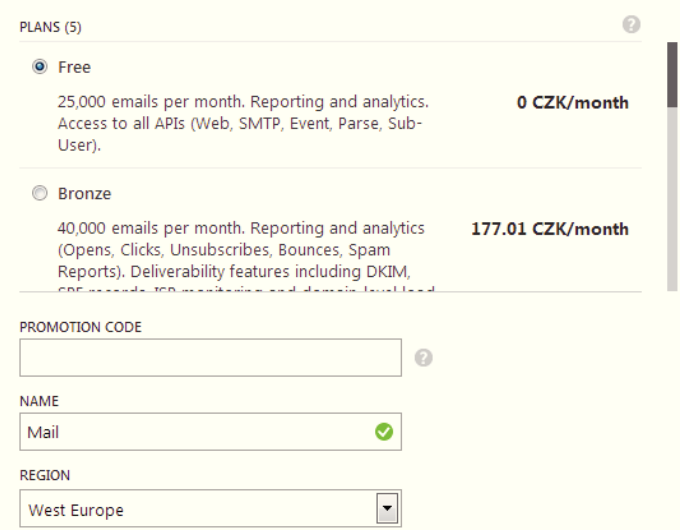
Po vyjetí seznamu různých služeb najdeme již zmíněný SendGrid. Po zvolení je nutné nastavit jaký typ služby chceme mít. Množství emailových zpráv a přístup k jednotlivým funkcím služby SendGrid je závislé na ceně. SendGrid lze využívat také bezplatně, ale dostaneme jen základní funkce a 25 tisíc emailových zpráv měsíčně.

Jak je možné vidět na obrázku 25 musíme si také zvolit jednoznačný název, který bude určovat danou službu a dále máme možnost si vybrat i datové centrum, které chceme využívat.

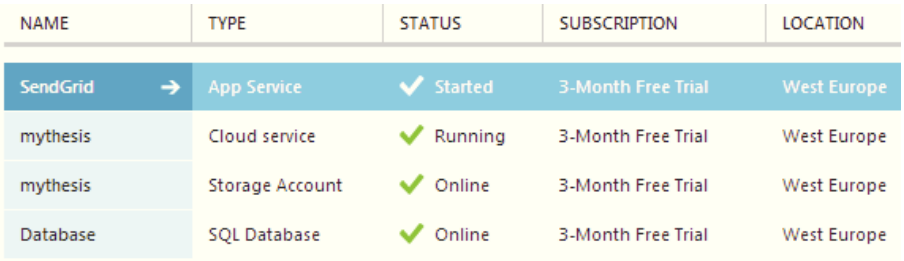
Po potvrzení se služba vytvoří a my ji můžeme vidět v přehledu všech služeb v menu All items (obrázek 25).

Abychom mohli používat SendGrid v aplikaci, je nutné do aplikace uložit přihlašovací údaje. Tyto údaje jsou v Connection info ve službě SendGrid. Tyto údaje jsou heslo, vygenerované uživatelské jméno a název smtp serveru(obrázek 26).

Tyto údaje pak uložíme v souboru web.config v sekci AppSettings (výpis kódu 18).



Obrázek 24: Výběr specifikace služby SendGrid



Obrázek 25: Běžící aplikace

```
<appSettings>
        <add key="emailName" value="
            azure_92e4200b75669c77dffa7202a163f5e5@azure.com"/>
    <add key="emailPass" value="acvqp0of"/>
    <add key="smtp" value="smtp.sendgrid.net"/>
</appSettings>
```

Výpis 18: Výpis prihlašovacích údaju pro aplikaci SendGrid

Poslední věc, která je nutná pro zasílání emailových zpráv přes aplikaci je volba domény, která se přijemcům zpráv bude zobrazovat. V našem případě to je doména forms.cs.vsb.cz. Tuto doménu lze opět nastavit v souboru web.config v sekci AppSettings(výpis kódu 19).

```
<appSettings>
        <add key="emailAdress" value="forms@vsb.cz"/>
</appSettings>
```

Obrázek 26: Přihlašovací údaje služby SendGrid

Výpis 19: Výpis domény pro aplikaci SendGrid

6.5 Nastavení pro přihlášení

I když aplikace využívá svoji databázi uživatelů, je možne využít již vytvořené účty u jiných aplikací a používat je k přihlašování. Tato možnost existuje u dvou typů. Jedním z nich je Facebook a druhým Windows Live.

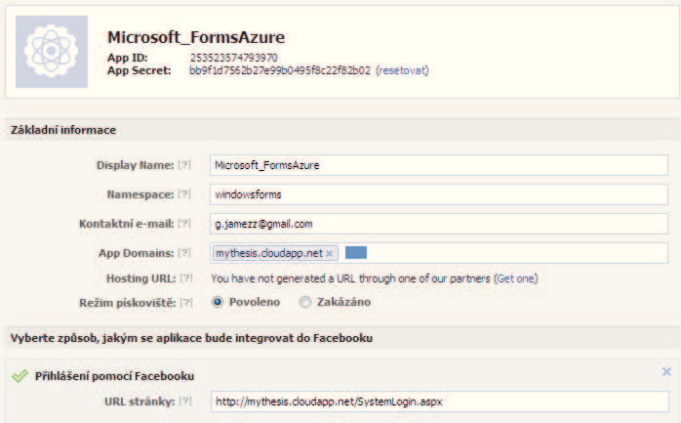
6.5.1 Nastavení pro připojení pomoci Facebook aplikace

Abychom mohli používat přihlášení pomocí Facebooku, je nutné mít aplikaci zaregistrovanou na adrese https://developers.facebook.com/. Zde se získá identifikační klíč, který se používá pro ověření pravosti aplikace. Druhý důležitý údaj je volba URL adresy, ze které bude požadavek na přihlášení pocházet(obrázek 27).

Identifikační klíč používá přímo javascriptový kód, který je zobrazen v příloze E. Tento klíč ulžíme v souboru web.config(výpis kódu 20)

```
<add key="FacebookAppID" value="162439323916426"/>
```

Výpis 20: Výpis prihlašovacích údaju pro Facebook ze sekce AppSettings



Obrázek 27: Registrace aplikace u facebooku

6.5.2 Nastavení pro připojení pomoci Windows Live aplikace

Pro používání přihlašovacích údajů pomocí Windows Live je nutné aplikaci zaregistrovat na stránkách https://manage.dev.live.com. Na těchto stránkách je nutné mít vytvořený svůj účet. V nabídce menu vybereme Create application a zadáme název aplikace a doménu, která bude používána pro ověřování uživatelů. Zde se také vygeneruje identifikační klíč, který se použije v aplikaci(obrázek 28).

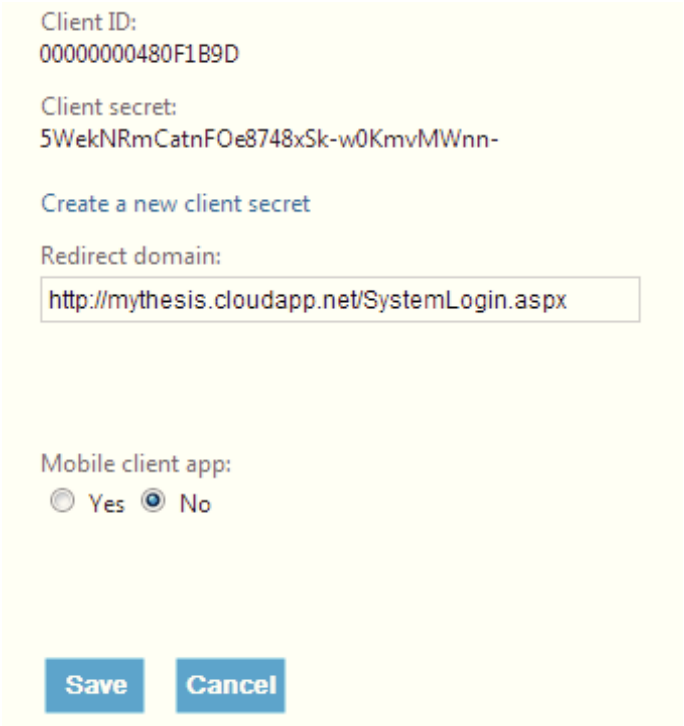
Po úspěšném zaregistrování aplikace musíme ještě do souboru web.config získané údaje zaznamenat. Do konfigurační sekce AppSettings ukládáme jak identifkační klíč aplikace, tak i doménu(výpis kódu 21). Celý javascriptový kód je zobrazen v příloze E.

```
        <add key="WindowsLiveAppID" value="00000000480F0224"/>
    <add key="WindowsLiveRedirectUrl" value="http://mythesis.cloudapp.net/SystemLogin.
        aspx"/>
```

Výpis 21: Výpis prihlašovacích údaju pro Windows Live ze sekce AppSettings

6.5.3 Administrátorský účet

Pouze účet, který je definován jako administrátorský, má přístup k funkcím jako je správa účtů, přehled zaznamenaných chyb apodobně. Automaticky je tento účet přístupný pod jménem admin a heslem admin. Heslo si pak vlastník aplikace může změnit dle libosti.



Obrázek 28: Registrace aplikace u Windows Live

7 Závěr

Cílem práce bylo vytvořit program, který by nejen dynamicky vytvářel formuláře pro různá setkání či přednášky, ale i pro potřeby jednoduchých dotazníků včetně anonymního dotazování. K vytvoření takové aplikace byla použita technologie ASP.NET a Windows Azure.

Technologie Windows Azure umožňuje, aby aplikace byla volně dostupná na internetu a zároveň nebylo třeba se starat o správu serveru, nebo jeho umístění. Cloudová datacentra nabízejí širokou škálu funkcí a prostředků pro správu aplikací a poskytují velmi různorodé typy úložišť, které mohou aplikace využívat. Různá úložiště mají své klady i zápory. Například lokální úložiště je sice velice výkonné, ale je bezestavové, takže si nemůžeme být jisti jestli tam ukládaná data zůstanou. Na druhou stranu úložiště blobu je sice pomalé při zpracovávání dat, ale k uloženým datům je velice jednoduchý přístup a to i přes HTTP rozhraní. K velkým nevýhodám cloudového úložiště patří práce s maily, kdy zasílání mailů podléhá přísným pravidlům a omezením z důvodu obavy ze zneužití.

Při řešení zadání jsem využil znalostí získaných v předmetech zabývajících se programováním v technologii .NET, ale i získáváním informací z odborných publikací a internetových článků, které se zabývaly danou problematikou.

Systém by bylo určítě možné více zintegrovat s různými sociálními sítěmi jako je například Facebook a více využít jejich funkce k vetší dostupnosti vytvořených formulářů.

8 Reference

[1] Juval Lowy, Working With The .NET Service Bus, MSDN Magazine, URL: https://www.windowsazure.com/en-us/home/features/overview/, Leden 2012 [e-článek]

[2] Neil Mackenzie, Overview of Windows Azure, URL: http://convective.wordpress.com/2011/04/03/overview-of-windows-azure/, Duben 2011 [e-článek]

[3] Microsoft, Overview of Windows Azure, URL: https://www.windowsazure.com/en-us/home/features/overview/, Leden 2012 [e-článek]

[4] Microsoft, Overview of Windows Azure, URL: https://www.windowsazure.com/en-us/home/features/storage/, Leden 2012 [e-článek]

[5] Microsoft, Overview of Windows Azure Connect, MSDN microsoft, URL: http://msdn.microsoft.com/en-us/library/windowsazure/gg432997.aspx, Prosinec 2011 [e-článek]

[6] Microsoft, Access Control Service 2.0, MSDN microsoft, URL: http://msdn.microsoft.com/en-us/library/windowsazure/gg429786.aspx, Prosinec 2011 [e-článek]

[7] Windows server 2012, Microsoft_Press_ebook_Introducing_Windows_Server_2012_PDF.pdf, URL: http://download.microsoft.com/download/0/C/B/0CB33133-C6F7-48A6-B7CC-D927988FCB32/Microsoft_Press_ebook_Introducing_Windows_Server_2012_PDF.pdf, Leden 2013 [Pdf dokument]

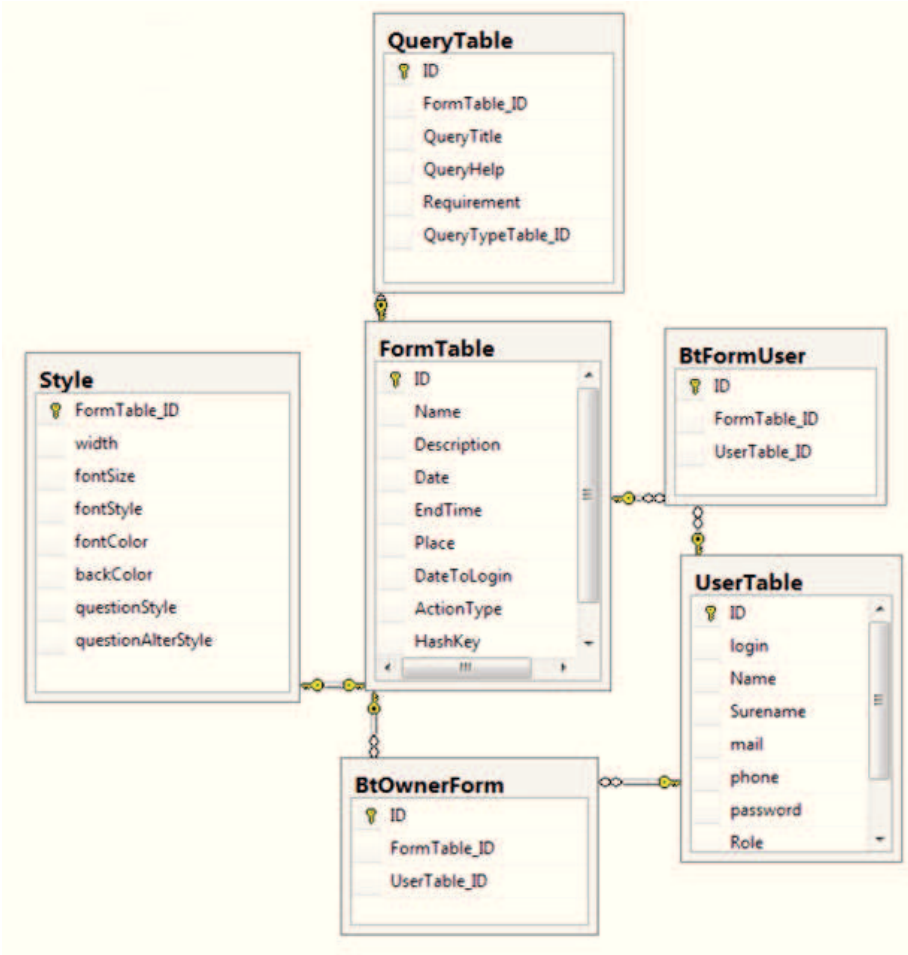
[8] Podrobný popis nasazení aplikace na cloud, dostupný z WWW adresy: http://download.microsoft.com/download/A/D/8/AD846E9E-CCEB-49D1-AC50-25103C3659A5/CzechAzureLabs.zip [zip-dokument]

A Vypis z Global.asax

```
public class Global : System.Web.HttpApplication
    {
        void Application_Start(object sender, EventArgs e)
        {


        }

        void Application_End(object sender, EventArgs e)
        {
            // Code that runs on application shutdown

        }

        void Application_Error(object sender, EventArgs e)
        {
            string url = HttpContext.Current.Request.Url.ToString();
            HttpContext.Current.Session["errorUrl"] = url;

            Server.Transfer("~/Error.aspx");

        }

        void Session_Start(object sender, EventArgs e)
        {
            // Code that runs when a new session is started

        }

        void Session_End(object sender, EventArgs e)
        {
            // Code that runs when a session ends.
            // Note: The Session_End event is raised only when the sessionstate mode
            // is set to InProc in the Web.config file. If session mode is set to StateServer
            // or SQLServer, the event is not raised.

        }
    }
```

Výpis 22: Výpis souboru Global.asax

B Schéma form table



Obrázek 29: Schéma FormTable

C Podrobný výpis chyby

Obrázek 30: Podrobný výpis chyby

D Výpis konfiguračního souboru web.config

```
        <appSettings>
            <add key="emailAdress" value="forms@vsb.cz"/>
            <add key="smtp" value="smtp.vsb.cz"/>
            <add key="FacebookAppID" value="1234567890"/>
          <add key="WindowsLiveAppID" value="9876543210"/>
          <add key="WindowsLiveRedirectUrl" value="www adresa"/>
      </appSettings>

      <connectionStrings>
    <add name="ConnectionString" connectionString="Server=tcp:wqshc5hzjm.database.windows.
        net,1433;Database=Database;User ID=Tester@wqshc5hzjm;Password=Diablolive123;
        Trusted_Connection=False;Encrypt=True;Connection Timeout=30;" />
    <add name="DatabazeConnectionString" connectionString="Data Source=la0t57h4l7.database.
        windows.net;Initial Catalog=Database;User ID=jamezz;Password=diabloSQL1"
        providerName="System.Data.SqlClient" />
    <add name="DbFormEntities" connectionString="metadata=res://*/Model1.csdl|res://*/Model1.
        ssdl|res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string=&amp;quot;
        data source=JAMEZZ-PC\SQLEXPRESS;initial catalog=DbForm;integrated security=
        True;MultipleActiveResultSets=True;App=EntityFramework&amp;quot;" providerName="System
        .Data.EntityClient" />
  </connectionStrings>

  <customErrors mode="On" defaultRedirect="~/ApplicationError.aspx">
      <error statusCode="404" redirect="~/Error404.aspx" />
    </customErrors>
```

Výpis 23: Úprava nastavení odchozích zpráv

E Obsah CD

- text práce;
- zdrojový kód systému;
- ukázková aplikace s vloženými Unit Testy.
- ukázková aplikace připravena pro nasazení na cloud